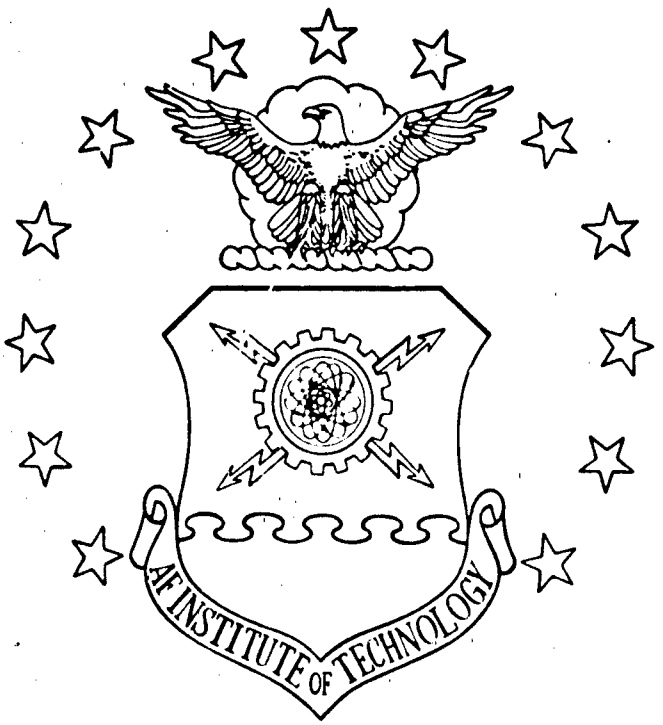


ADAPTIVE-GRID OPTIMIZATION FOR MINIMIZING STEADY-STATE, TRUNCATION ERROR

THESIS

Donald E. Coffey, Jr.

Captain, USAF

AFIT/GA/AA/85D-4

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GA/AA/85D-4

ADAPTIVE-GRID OPTIMIZATION FOR MINIMIZING STEADY-STATE, TRUNCATION ERROR

THESIS

Donald E. Coffey, Jr.

Captain, USAF

AFIT/GA/AA/85D-4

Approved for public release distribution unlimited

AFIT/GA/AA/85D-4

ADAPTIVE-GRID OPTIMIZATION FOR MINIMIZING STEADY-STATE, TRUNCATION ERROR

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Astronautical Engineering

Donald E. Coffey, Jr., B.S.

Captain, USAF

December 1985

Preface

This study shows that optimization techniques can be applied to the solution of complex, boundary-layer flow problems. It builds on previous work done by students and faculty at the Air Force Institute of Technology. Captain Karen Lange working with Major James K. Hodge developed a computer code to solve boundary-layer problems. Captain Lange's computer code was modified, and optimization codes studied by 1st Lieutenant Bruce K. Boyd and Captain Sal A. Leone were added. Without the work of these individuals, my extension of both studies to optimization of two-dimensional, boundary-layer code would not have been possible.

I would like to take this opportunity to especially thank my advisor, Dr. Sal A. Leone, for his guidance, knowledge, and assistance throughout this study. Of course, this study could not have proceeded very far, without the knowledge and expertise of Major Hodge in Computer Fluid Dynamics. I thank him for always willingly sharing his expertise when I needed it. Lastly, I thank Lt. Col. Eric Jumper and Dr. M. L. Rasmussen for giving me the foundation in theoretical, fluid dynamics that enabled me to understand the physics of the flow problem. If in some small way, this study helps further studies of similar problems or advances the study of hypersonic, fluid dynamics, it is a success.

Donald E. Coffey, Jr.

Table of Contents

List of Figures

List of Tables

List of Symbols

Symbol	Description
h	heat convection coefficient
hs	specific enthalpy
i	streamwise difference coordinate
j	normal difference coordinate
k	heat conduction coefficient
p	pressure (lb/ft^2)
q	heat flux
r	Iteration level(Powell's method)
t	physical and transformed time coordinate
u	streamwise velocity component-physical plane
v	normal velocity component-physical plane
x	streamwise position coordinate-physical plane
y	normal position coordinate-physical plane
A	surface area (ft^2)
B	velocity profile coefficient
E	Direction coordinate-(Powell's method)
F	Minimized function (Powell's method)
H	Total Enthalpy
L	surface length (ft)
M	Mach number
P	Poisson equation streamwise control function
Q	Poisson equation streamwise control function
R	Gas constant- 1715 slug-sec/ft ^{o}R

Symbol	Description
U	streamwise velocity component- transformed plane
V	normal velocity component-transformed plane
X	streamwise position coordinate-transformed plane
Y	normal position coordinate-transformed plane
β	shock angle
γ	gas specific heat= 1.4
δ	boundary layer thickness- physical plane
δ_t	thermal boundary layer thickness
η	normal position coordinate- computational plane
θ	wedge deflection angle
λ	Powell's method direction parameter
μ	fluid viscosity (slugs/ft-sec)
ξ	streamwise position coordinate-computational plane
ρ	density (slugs/ft^3)
τ	time-computational plane
Ω	von Karman-Pohlhausen shape factor
Δ	boundary layer thickness- transformed plane
Σ	summation
Re	Reynold's number-($\rho_\infty u_\infty L/\mu_\infty$)
C_f	skin friction coefficient
Pr	Prandtl number-($Cp\mu/k$)
Cp	specific heat at constant pressure-(6006 $ft^2/s^{2o}R$)

Subscripts

e	edge conditions
∞	freestream conditions
w	surface, or wall, conditions

Symbol	Description
	Subscripts
ref	reference conditions
t	theoretical solutions
	Superscripts
'	dimensional variable

Abstract

This study develops an adaptive-grid method which minimizes the truncation error in the finite-difference solution. The study solves compressible, steady-state, boundary-layer equations assuming perfect-gas flow over an isothermal wall. The Dorodnitsyn, compressibility transformation changes the boundary-layer equations, as expressed in two-dimensional, cartesian coordinates, into an incompressible form. The equations are then transformed into variables of a computational plane. Implicit Successive-Over-Relaxation (SOR) solves the finite-differenced, computational, boundary-layer equations. Comparison of the computed solution for incompressible flow over a flat plate to Blasius', exact solution shows the boundary-layer code is accurate.

The adaptive-grid method uses Powell's method to optimize the solution grid by minimizing the sum of the third derivative in the computational plane of the tangential velocity component. Powell's method finds the grid, control function, Q, in an elliptic, grid equation, $Y_{\eta\eta} + QY_{\eta}=0$, which minimizes a specified function. The grid equation generates the grid spacing at the end of the plate. This spacing is then streamwise scaled across the remaining grid. Minimizing the sum of the square of the third derivative in the computational plane of the tangential velocity component, $U_{\eta\eta\eta}{}^2$, over the entire domain decreases the truncation error

the best of the functions tested. This study tests the sum of the squares of the first, second, and third derivative of the tangential velocity as minimized functions. The accuracy of the optimized, adaptive-grid solution is greater than the original, fixed-grid solution. The study then applies the optimization to supersonic and hypersonic problems. The computed, adaptive-grid solutions show good correlation with theoretical models for supersonic and hypersonic flow developed by Van Driest and Eckert.

ADAPTIVE-GRID OPTIMIZATION FOR MINIMIZING STEADY-STATE, TRUNCATION ERROR

Chapter I: Introduction

Every flight of the Space Shuttle highlights the knowledge aeronautical engineers have of hypersonic flight. The success of each flight shows a basic understanding of the problems involved with flight at high altitudes and at speeds above Mach 5. However, if future aeronautical engineers are to build better shuttles or to develop a transatmospheric vehicle, increased understanding of hypersonic flight will be necessary. Presently, experimentation in the wind tunnel and Space Shuttle flights validates portions of the hypersonic theory. Unfortunately, experimental work of this sort is very expensive. Newer methods of modeling flow using computers offer hope for simulating characteristic, fluid properties in the hypersonic regime without the experimental expense. Using computational fluid methods, unexpected experimental results can be confirmed, and, conversely, experiments can be designed to validate results from computational fluid modeling. In this way, better understanding of hypersonic fluid behavior can be gained and can lead to more efficient hypersonic vehicles.

An example of the usefulness of computational modeling for explaining the results of experimental data is the non-

isothermal wall effects which have occurred on the Space Shuttle. The surface of any body travelling at high speeds through any fluid heats up. The effect the heated surface has on the flow field or boundary layer around the venicle depends on the heat transfer coefficient of the surface. If the body is one material and is heated evenly, the convection which takes place is isothermal. However, if, as is the case with the Space Shuttle, there are several different materials joined together on the same surface, non-isothermal convection o curs. Non-isothermal convection can change the air and heat flow on the vehicle. Boundary-layer theory for the hypersonic, non-isothermal wall predicts a discontinuity in wall temperature in the area of a material change (11). However, wind tunnel data suggests a much slower, wall-temperature recovery than theory suggests (21:2). Roberts and Lange, in two separate studies, investigated the non-isothermal-wall effect using computational fluid dynamics (CFD) (15,21). Roberts modeled flows with two-dimensional Navier-Stokes equations. Lange modeled flow using an unsteady, boundary-layer analysis. Both studies showed a non-isothermal, wall effect. However, in the case of the boundary-layer analysis, inaccuracies in the method tended to smear the quantitative results. The disadvantage with most computer, modeling techniques is truncation error and round-off error smear the results in high-gradient areas. These high-gradient areas also are the parts of the f.ow where the non-isothermal, wall effect takes place. Therefore, the engineer

must decrease the errors in the high-gradient areas to more accurately model flow conditions. In this way, computer fluid modeling can more accurately predict flow conditions such as the non-isothermal, wall effect, which otherwise would not be seen.

Background

Using an adaptive grid which is changed after each solution of the modeling equations is a way to increase the accuracy of the computer solution without increasing the number of solution, grid points. The easiest way to increase resolution of regions where fluid properties, like temperature and pressure, are changing rapidly is to increase the number of grid points. The other way is to use an adaptive grid for solving the model equations. The adaptive grid concentrates grid points in high-gradient areas, (i.e. boundary layers), to increase resolution and spreads out grid points in low-gradient regions. This keeps the total number of grid points small, but places grid points to produce the best resolution. For a time-dependent problem, Ghia, Ghia, and Shin develop a flow-dependent, adaptive-grid method to increase the accuracy of the computer solution. This adaptive-grid method reforms the solution grid after each time iteration of the modeling equations by minimizing the magnitude of the convective terms of the governing equations which indirectly minimizes the truncation error in the governing equations (7:36-37). Their treatment of the two-dimensional, boundary-

layer equations adapts the grid in normal and streamwise directions. However, since the gradients in a boundary layer are not very large in the streamwise direction, it may be possible to optimize the solution grid in only the normal direction and still get an accurate solution. If the particular, flow solutions do not have similar, velocity profiles, the solution grid can be coupled with a parabolic-grid generation method (9). Regardless, one-dimensional, grid, optimization techniques can be applied to optimize the grid in the normal direction. Leone and Hodge suggest optimizing the adaptive grid in one dimension with Powell's minimization method (16). Boyd also uses a method to optimize the adaptive grid in one dimension. Using a least squares curve fit to model a quadratic equation and a Newton-Raphson optimization of a grid control function, Q, Boyd attempts to minimize the truncation error of the third derivative of the dependent variable to get a more accurate one-dimensional, flow solution (3:11). After optimizing the solution grid in the normal direction with one-dimensional techniques, the characteristics of the boundary layer solution can be used to scale the normal optimization in the streamwise direction. The resulting optimized, solution grid generates more accurate flow solutions.

Objective

This thesis develops a method for optimizing, a steady-state, adaptive-grid solution of two-dimensional, flow prob-

lems by minimizing the truncation error in the streamwise velocity component. The adaptive grid minimizes the sum of the square of the third derivative of the streamwise velocity in the transformed plane, $\sum U_{\eta\eta\eta}{}^2$, using Powell's minimization method. Powell's method iterates on an n-dimensional array using conjugate directions until finding the minimum of the function without calculating the derivatives of the function on that array. In this study, Powell's method primarily minimizes the third derivative of the streamwise velocity component by optimizing the grid control function, Q, in the equation, $Y_{\eta\eta}+QY_{\eta}=0$. Because of the flexibility of Powell's method, the effect of optimizing the solution grid by minimizing the first, second, or a combination of all three streamwise velocity derivatives is checked to verify truncation error arguments. Regardless of the specific minimized function used, the method optimizes the grid in the normal direction. The grid is then scaled in the streamwise direction using boundary-layer theory. A modification of Lange's boundary-layer code then solves the flow problem. Comparing the incompressible and compressible flow solutions to exact solutions and Lange's non-adaptive numerical solutions verifies the accuracy of the adaptive-grid solution. The increase in accuracy with the optimized, adaptive grid should be evident in the velocity calculations and in the comparison of heat transfer along the surface.

Chapter II: Theory

Boundary-Layer Equations

The complete Navier-Stokes equations predict fluid characteristics for all flow conditions (23:64). This set of equations is however very difficult to solve. Since the majority of viscous effects and heat transfer effects appear in a thin layer adjacent to the flow surface, this study investigates only the portion of flow in that region, known as the boundary layer. Flow properties in the boundary layer allow several simplifying assumptions which result in a simpler set of flow, modeling equations than the Navier-Stokes equations. For high Reynold's number flows, viscous effects in the form of shearing stress at the wall, $\tau_w = \mu(\partial u/\partial y)_{y=0}$, and velocity gradients in the normal direction, $\partial u/\partial y$, are very large inside the boundary layer. Outside the boundary layer, flow is essentially inviscid with negligible, velocity gradients (23:128-129). Therefore, within the boundary layer thickness, δ, it is assumed that:

a. gradients in the normal directions are much larger than gradients in the streamwise direction,

b. the normal velocity, v, is much smaller than the streamwise velocity, u, and

c. all terms of order δ/L or smaller are negligible.

For laminar flow over a flat plate or wedge, the pressure gradient in the streamwise direction, $\partial p/\partial x$, of the inviscid

region is impressed on the boundary layer. This pressure gradient is assumed to be zero for a flat plate parallel to the flow. Further, the boundary layer equations are non-dimensionalized using the following relationships.

$$x = \frac{x'}{L'}, \quad y = \frac{y'}{L'}, \quad t = \frac{t'U'_\infty}{L'}, \quad u = \frac{u'}{U'_\infty}, \quad v = \frac{v'}{U'_\infty}, \tag{1}$$

$$H = \frac{H'}{U'^2_\infty}, \quad T = \frac{C_p'T'}{U'^2_\infty}, \quad p = \frac{p'}{\rho_\infty' U'^2_\infty}, \quad \mu = \frac{\mu'}{\mu_\infty'}, \quad \rho = \frac{\rho'}{\rho_\infty'}$$

The resulting non-dimensionalized, boundary-layer equations for compressible, laminar flow are:

Continuity:
$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} = 0 \tag{2}$$

Momentum:
$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2}{\partial x} + \frac{\partial \rho uv}{\partial y} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial y}\frac{\mu}{Re}\frac{\partial u}{\partial y} \tag{3a}$$

$$\frac{\partial p}{\partial y} = 0 \tag{3b}$$

Energy:
$$\frac{\partial \rho H}{\partial t} + \frac{\partial \rho uH}{\partial x} + \frac{\partial \rho vH}{\partial y} = \frac{\partial p}{\partial t} + \frac{\partial}{\partial y}\frac{\mu}{RePr}\frac{\partial H}{\partial y} + \frac{\partial}{\partial y}\frac{\mu(Pr-1)}{RePr}\frac{\partial u^2/2}{\partial y} \tag{4}$$

Boundary conditions for the boundary layer equations are $u' = v' = 0$ at $y' = 0$ and $u' = U_e$ at $y' = \delta$. Assuming incompressible flow or transforming the compressible equations into an incompressible form using the Dorodnitsyn transformation further simplifies the above equations (22:101).

Incompressible, Boundary-Layer Theory

If the flow density is constant, the velocity profile,

as well as the heat transfer characteristics near the wall, have already been derived theoretically for both exact and approximate cases. In the incompressible case, the boundary layer equations are:

Continuity:
$$\frac{\partial U}{\partial X}+\frac{\partial V}{\partial Y} = 0 \tag{5}$$

Momentum:
$$\frac{\partial U}{\partial t}+U\frac{\partial U}{\partial X}+V\frac{\partial U}{\partial Y} = -\frac{1}{\rho}\frac{\partial p}{\partial X}+\frac{\partial}{\partial Y}\left(\frac{\mu\rho}{Re}\frac{\partial U}{\partial Y}\right) \tag{6a}$$

$$\frac{\partial p}{\partial Y} = 0 \tag{6b}$$

Energy:
$$\frac{\partial H}{\partial t}+U\frac{\partial H}{\partial X}+V\frac{\partial H}{\partial Y} = \frac{1}{\rho}\frac{\partial p}{\partial t}+\frac{\partial}{\partial Y}\left(\frac{\mu\rho}{RePr}\frac{\partial H}{\partial Y}\right) + \frac{\partial}{\partial Y}\left(\frac{\rho\mu(Pr-1)}{RePr}\frac{\partial U^2/2}{\partial Y}\right) \tag{7}$$

where $u=U$, $v=V$, $x=X$, and $y=Y$

Blasius developed the exact solution to incompressible, boundary layer flow over a flat plate. The results of his exact solution define the thickness of the boundary layer. If the edge of the boundary layer is assumed to be where U'/U_e is .994, the boundary layer thickness, δ, at any streamwise location, x', is

$$\delta(x') = \frac{5.2x'}{\sqrt{Re_x}} \tag{8}$$

where $Re_x=\rho_\infty' u' x'/\mu'_\infty$. Blasius' solution also gives an exact value for the skin friction coefficient, C_f.

$$C_f(x') = \frac{0.664}{\sqrt{Re_x}} \tag{9}$$

The computer solution of the incompressible, flat, plate problem should match this exact C_f result.

Though Blasius' incompressible, flat, plate solution is exact and is the best description of this velocity profile, the von Karman-Pohlhausen, approximate solution of the two-dimensional, flow problem offers more easily obtainable velocity profile results. The von Karman-Pohlhausen method solves the momentum-integral equation (23:160). The velocity distribution is a fourth-order polynomial where $\omega = y'/\delta(x)$.

$$U = \frac{U'}{U_e} = a\omega + b\omega^2 + c\omega^3 + d\omega^4 \tag{10}$$

The four boundary conditions required to solve the coefficients in Eq (10) are

$$y' = 0 : \quad U = 0 \qquad \frac{\partial^2 U}{\partial Y^2} = \frac{1}{\rho}\frac{\partial p}{\partial X} = -U_e \frac{dU}{dX}_e$$

$$y' = \delta : \quad U = 1 \qquad \frac{\partial U}{\partial Y} = 0, \quad \frac{\partial^2 U}{\partial Y^2} = 0 \tag{11}$$

The coefficients, a-d, in Eq (10) are

$$a = 2+\Omega/6 \;, \quad b = -\Omega/2 \;, \quad c = -2+\Omega/2, \quad d = 1-\Omega/6$$

$$\text{where} \quad \Omega = \delta^2 \frac{\mu}{\rho}\frac{dU}{dX}_e \tag{12}$$

(23:207). Ω is the Pohlhausen pressure, gradient parameter. For flat plate and wedge flows with no streamwise, pressure gradient, Ω is zero. The von Karman-Pohlhausen solution provides another verification for numerically-derived, velocity profiles in this study. After verifying the accuracy of

the boundary-layer and adaptive-grid code with the Blasius or von Karman-Pohlhausen results, then compressible forms of the boundary-layer equations are solved.

Compressibility Considerations

Since the incompressible, boundary-layer equations, Eqs (5), (6), and (7), already have approximate and exact solutions, the solution of the compressible, boundary-layer equations, Eqs (2), (3), and (4), is possible if the change in density and viscosity across the boundary layer can be accounted for. The Dorodnitsyn transformation changes the compressible equations into an incompressible form. The compressible equations can then be solved as incompressible equations. The Dorodnitsyn transformation is "a nonlinear stretching of the normal coordinate" (20). The transformation takes the normal coordinate, y, in the physical space and stretches it into a new normal coordinate, Y, in the transformed space. The transformed, space coordinates are now expressed in X and Y where

$$X = x, \qquad Y = \int_0^y \frac{\rho'}{\rho_e'}\, dy \tag{13}$$

The transformed, grid coordinates lead to a change in the velocity components, as shown in Appendix A. The transformed, velocity components are

$$U = u, \qquad V = \rho v + \partial Y/\partial t + u\partial Y/\partial x \tag{14}$$

Using the transformed variables in Eqs (13) and (14), changes

the compressible, boundary-layer equations, Eqs (2) - (4), into the incompressible form in Eqs (5) - (7). The boundary layer thickness, $\delta(x)$, also changes into a transformed, boundary-layer thickness, $\Delta(X)$. Assuming a calorically-perfect gas, the density ratio, ρ'/ρ'_e, equals the inverse of the specific enthalpy ratio, hs_e'/hs', and

$$\delta(x) = \int_0^{\Delta(X)} \frac{hs'}{hs_e'} dY \qquad (15)$$

An explicit expression for $\Delta(x)$ comes from an integral analysis of the new, boundary-layer equations (20).

$$\Delta(X) = \sqrt{\frac{\rho'_w \mu_w' * 2B*X'}{\rho'_e U_e * f_\theta * (2\sigma + 1)}} \qquad (16)$$

where $B = \dfrac{dU/U_e}{d(Y/\Delta)}$

and $f_\theta = \int_0^1 U/U_e(1-U/U_e)\, d(Y/\Delta)$

σ, in the above equation, is 0 for two-dimensional flow and 1 for axisymmetric flow (Appendix F). Assuming flow over the flat plate or wedge develops a cubic, velocity profile such that

$$U/U_e = 1.5(Y/\delta) - 0.5(Y/\delta)^3 , \qquad (17)$$

the value of $2\ B/f_\theta$ is 280/13. Known flow conditions then determine $\Delta(X)$. After solving the flow for the incompressible form, inverse transformations compute the solution in

the physical space. The inverse transformations result from integrating the computed, enthalpy ratio to recover the physical-grid, normal coordinate, or

$$y = \int_0^Y \frac{hs'}{hs_e'} \, dY \tag{18}$$

The compressible solution of the boundary layer equations in the physical plane for a given set of flow conditions is then complete.

Flow Properties

Oblique, Shock-Wave Theory.

For supersonic flow over a flat plate, the flow conditions such as density, pressure, temperature, and velocity remain relatively constant outside the boundary layer. However, if the flat plate is inclined to the direction of flow to simulate a wedge, the flow conditions change as flow passes through the shock wave which develops in front of the wedge. This study assumes a perfect gas flow with no dissociation of the gas molecules. For the perfect gas,

$$p' = \rho' R\, T', \text{ and } \quad \rho = \frac{\gamma}{\gamma - 1} \frac{p}{T} \tag{19}$$

where the gas constant, R, is 1715 ft-lb/slug-^{o}R and the specific heat ratio, γ, is 1.4. Oblique, shock-wave theory applies to this flow problem. To find the conditions behind the shock wave, the shock angle, β, must be found. Liepmann and Roshko use an explicit form of the shock-angle equation

(17:87). However, the implicit form,

$$M_e^2 \sin^2\beta - 1 = \frac{\gamma+1}{2} M_\infty^2 \frac{\sin\beta\sin\theta}{\cos(\beta-\theta)} \qquad (20)$$

is more useful for iteratively computing the shock angle, β, given a deflection angle, θ, and a freestream, Mach number, M_∞. With the shock angle and freestream values of Mach, M_∞, pressure, p_∞, and temperature, T_∞, the equations

$$M_e^2 \sin^2(\beta-\theta) = \frac{2 + (\gamma-1) M_\infty^2 \sin^2\beta}{2M_\infty^2 \sin^2\beta - (\gamma-1)} \qquad (21)$$

$$p_e = p_\infty\left[1 + \frac{2\gamma}{\gamma+1}(M_\infty^2 \sin^2\beta - 1)\right] \qquad (22)$$

$$T_e = T_\infty\left[1 + \frac{2(\gamma-1)}{(\gamma+1)^2} \frac{M_\infty^2 \sin^2\beta - (\gamma M_\infty^2 \sin^2\beta + 1)}{M_\infty^2 \sin^2\beta}\right] \qquad (23)$$

define the edge conditions behind the shock wave for Mach number, pressure, and temperature (17:86). All other edge conditions such as enthalpy and density can be found from these quantities. These edge conditions affect the solution of the boundary-layer equations over the given surface.

Viscosity.

Viscosity is not a constant across the boundary layer. Since viscosity is a function of the fluid temperature which changes across the boundary layer, viscosity also changes. Generally, engineers use Sutherland's, viscosity law,

$$\mu' = \frac{2.2685 \times 10^{-8}\, T'^{3/2}}{T' + 198.6\ ^\circ R} \quad \frac{\text{slugs}}{\text{ft-sec}} \qquad (24)$$

for calculating viscosity as the fluid temperature changes.

However, Fiore suggests Sutherland's law is not valid for the low temperatures resulting from hypersonic, flow experiments (6:56). Flow temperatures in hypersonic, wind-tunnel testing generally range from 30°R to 200°R (6:56). A more, exact expression for low-temperature viscosity applies. Keyes obtains better low-temperature calculations of viscosity using the relation

$$\mu' = \frac{2.32 \times 10^{-8} T'^{1/2}}{1+[220/T'(10^{9/T'})]} \quad \frac{\text{slugs}}{\text{ft-sec}} \qquad (25)$$

(13). In addition, this low-temperature, viscosity law gives virtually the same results for temperatures above 300°R. Cappelano states "variation between the alternate form and Sutherland's law is negligible above a temperature of T= 300°R " (4:20). Below 300°R, there is a difference. The new, viscosity law is more accurate for low temperatures. Therefore, unless computational results must be compared with previous studies which use Sutherland's law, this study uses Keyes' viscosity law where appropriate.

Convective Heat Transfer Coefficient.

Convection is the primary method of heat transfer across the boundary layer. For a constant temperature surface, the heat flow, q, is calculated by the convective, heat relation

$$q = hA(T_w - T_{aw}) \qquad (26)$$

where T_w is the wall temperature, T_{aw} is the adiabatic, wall temperature, and A is the wall surface area. Thus, it is not

surprising that the ability to accurately compute the convective, heat, transfer coefficient, h, is important to any boundary-layer study. Numerically, the computer code calculates the heat flow rate per unit area from the flow solution using

$$q/A = -k\,\partial T'/\partial y' \tag{27}$$

where k is the thermal, conductivity coefficient for a fluid. The wall temperature, T_w is specified. T_{aw} varies with Mach number and fluid temperature by the adiabatic, wall temperature,

$$T_{aw} = T_e[1+(Pr)^{1/2}\ (\tfrac{\gamma-1}{2})M_\infty{}^2] \tag{28}$$

The Prandtl number is assumed constant and is

$$Pr = C_p\mu/k \tag{29}$$

The factor, $Pr^{1/2}$, in Eq (28) is the adiabatic, recovery factor for laminar flows (12:213). In assuming an adiabatic wall, Schetz explains,

> the temperature that the wall attains at equilibrium will depend on how much of this kinetic energy [kinetic energy of the flow] is recovered on the wall. This is expressed in the recovery factor... (22:5)

Solving Eq (26), using Eqs (27)-(29), gives a numerical solution for h. There is also a theoretical solution for h. The theoretical solution comes from Blasius' solution of the flat-plate problem as well as Eckert's, flat-plate theory. From Blasius', exact solution for flat plate flow, Eq (9) gives a value for C_f. C_f is also related to the Stanton

number, St_x, by

$$C_{fx} = 2St_x Pr^{2/3} \tag{30}$$

and

$$St_x = \frac{h_x'}{\rho' C_p U_\infty} \tag{31}$$

(12:195). Substituting Eq (31) into Eq (30), equating the result to Eq (9), and solving for h gives a theoretical value of h:

$$h_x' = .332 \frac{Cp}{Pr^{2/3}} [\mu_e' \rho_e' u_e'/x']^{1/2} \tag{32}$$

This value of h is only for flows where viscosity and density are constant, the wall is isothermal, and the fluid is a perfect, single-species gas with no dissociation of molecules. It applies therefore to incompressible flows treated in this study. However, for compressible, boundary-layer flows, where density and viscosity vary, the theory must be modified slightly. Eckert suggests the use of a reference temperature which is representative of the temperatures across the boundary layer in the previous constant property relations (22:96). Defined as

$$T^{*}' = T_e' + .5(T_w' - T_e') + .22(T_{aw}' - T_e') \tag{33}$$

the reference temperature is used to calculate reference values of viscosity and density, μ^{*}' and ρ^{*}', respectively. Then, Stanton number and h for compressible flows are calculated using the reference, *, conditions. Another reference

enthalpy, h_{ref}, is calculated by using μ_∞, ρ_∞, and u_∞ in place of the edge conditions in Eq (32). To evaluate how h changes from its freestream value, the theoretical h values for the incompressible and compressible cases are compared to the freestream h. For incompressible flow over a flat plate, with edge conditions equal to freestream conditions, h/h_{ref} should be one for both numerical and theoretical values of h. How close the numerical h/h_{ref} is to the theoretical h/h_{ref} indicates how well the computed solution models the exact, incompressible solution.

Theoretical Limitations

The assumptions which form the theoretical basis for the boundary-layer theory of this study pose several limitations on the general application of the solutions for all flow conditions in an experimental setting. The flows are assumed to be perfect gases. This limits the results to air flows whose temperatures and pressures guarantee negligible, rarefied-gas effects, i.e. molecular vibration or dissociation. Also, the requirement of a zero streamwise pressure gradient across the boundary layer limits the shapes to which it can be applied without modification. These shapes include flat plates, wedges, and cones. Most important is a limitation on the physical effects which can be predicted in the hypersonic regime. Boundary-layer theory is incapable of predicting the effects of shock-boundary-layer interaction. The shock wave produced at the leading edge of a sharp-nosed

body, like a wedge, is relatively far away from the surface and the boundary layer at low Mach numbers except in the immediate vicinity of the point (Fig. 1). However, at hypersonic Mach numbers, the shock wave comes close to the surface (Fig.2). In explaining his experiments on shock interaction, Nagamatsu comments,

> The shock wave and boundary layer seem to be merged before separating into distinct shock wave and boundary layer. As the flow Mach number was increased, the merged region extended over a larger portion of the plate(18:461).

He also points out some of the variations in flow properties this interactions causes.

> When the shock wave and the boundary layer were merged, the pressure from the shock wave to the surface for a given location from the leading edge was approximately constant. After the shock wave became separated from the boundary layer, the pressure behind the shock wave was greater than the surface pressure at the same location (18:462).

Boundary-layer theory is not able to predict any of these interactive results. Therefore, any solution at the leading edge of the body probably has an inherently, large error. This error at the leading edge should be ignored to get an indication of the overall error of the boundary-layer code over the surface. The finite-difference method also has problems predicting the large gradients at the leading edge. Therefore, neglecting large leading edge errors due to the boundary layer solution also leads to neglecting a large leading error due to the finite-difference method.

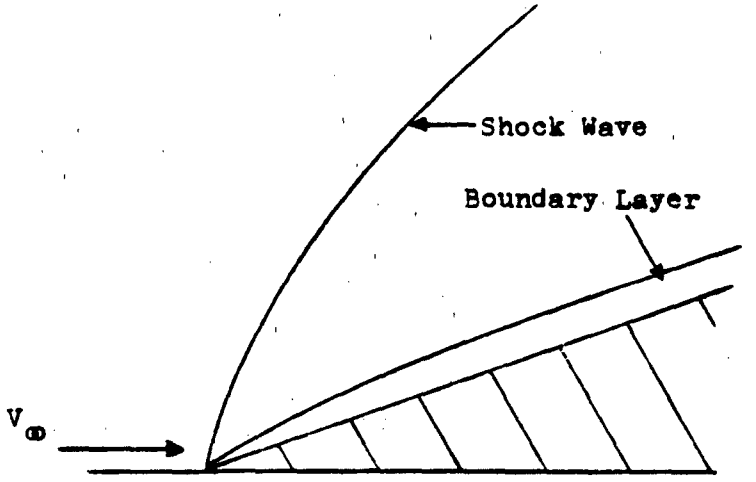


Fig. 1 Supersonic Shock-Boundary Layer Interaction

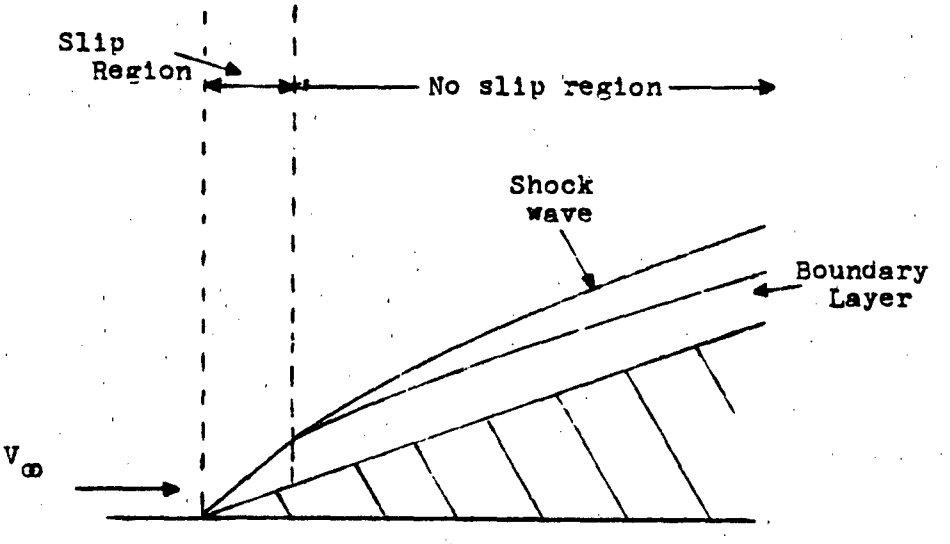


Fig. 2 Hypersonic Shock-Boundary Layer Interaction

Chapter III: Finite Difference Solution

The boundary-layer equations, Eqs (2) -(5), are a non-linear set of partial differential equations which describe fluid flow close to a surface. Their mathematical solution is difficult. However, using finite-difference techniques, the solution of the boundary-layer equations and the resulting description of the flow field is possible. First, the domain of the flow field must be divided, or differenced, into a grid pattern. This grid may be in any form which makes solution of the problem easiest. For this study, three different grids which characterize three different, solution spaces are of interest: the physical grid, the Dorodnitsyn-transformed grid, and the computational grid. The physical grid is an orthogonal, surface-normal grid which describes the flow pattern in the physical, compressible plane. In the physical plane, the boundary-layer equations are highly non-linear and vary with viscosity and density. Therefore, the Dorodnitsyn transform is introduced to take the density dependence out of the problem. This transformation adjusts the normal coordinate, y, to put the density dependence into the definition of a new normal coordinate, Y, instead of in the boundary-layer equations. This creates coordinates, X and Y, for the Dorodnitsyn-transformed grid. The final step is to take all non-linearity out of the grid by transforming X and Y to computational variables, ξ and η. Expressing the boundary-layer equations in terms of the new, computational varia-

bles results in solution equations which are more easily solved in a uniform, computational space. The computational grid with coordinates, ξ and η, has a constant, step size of one in both the streamwise and normal directions. The physical and Dorodnitsyn grids have varying step sizes. Boundary conditions and initial conditions are then added to the solution equations, and the equations are solved in the computational space using an implicit, Successive-Over-Relaxation (SOR) method.

The Solution Grid

The solution grid divides the flow field into exact, coordinate locations. The solution of the flow problem describes the velocity, enthalpy, and other flow conditions at the grid locations. For each flow problem, there is an optimum grid which best resolves the flow field. The accuracy of the solution reflects how well the grid determines the gradients in the flow field. With very large spacing between grid points, rapidly changing flow conditions in a particular area may not appear due to lack of resolution on the grid. Increasing the number of grid points increases resolution but also increases computational time. It is also a crude, brute force approach to the problem. An ideal solution is to put a large number of grid points in the high-gradient regions and fewer points in low-gradient regions without increasing the total number of grid points. The boundary-layer problem needs the majority of the points in the boundary layer, es-

pecially near the leading edge, where flow conditions are changing rapidly. The least concentration of points should be in the inviscid-flow region outside the boundary layer where conditions are essentially constant. The particular way this type of grid is built depends on the grid equation chosen.

This study uses a one-dimensional, grid equation to determine the grid spacing in one dimension, analagous to a multi-dimensional Poisson equation (an elliptic, partial differental equation). The other dimension is then scaled using a velocity profile of Blasius', boundary-layer solution. The one-dimensional grid equations are

$$X_{\xi\xi} + PX_{\xi} = 0 \quad \text{(streamwise direction) or} \tag{34}$$

$$Y_{\eta\eta} + QY_{\eta} = 0 \quad \text{(normal direction)} \tag{35}$$

where

$$\xi = \xi(x,y,t)$$

$$\eta = \eta(x,y,t) \tag{36}$$

P and Q are control functions. The grid equations stretch the grid points depending on the control functions P and Q. The grid used to verify results for Lange's non-adaptive grid boundary-layer program defines the points in the streamwise direction first. Setting P equal to zero forms a constant spacing in the x direction. Then, using a parabolic velocity profile, $U/U_e = (Y/\delta)^{1/2}$ for δ defined by Eq (8), determines the y coordinates. Due to the constant, x spacing and the parabolic characteristics of the power law, the power-law

grid is several, expanding, parabolic curves, as Fig. 3 shows. The flow solution which comes from using this power-law grid can be compared with the optimized grid solution. The optimized-grid solution also uses constant spacing in the x direction. However, it uses Eq (35) to fix the y coordinates at an x location and then scales the rest of the grid using Eq (8). Unified Difference Relations (UDR) given by Hodge, Leone, and McCarty solve the grid equation (9). The UDR for Eq (35) are

$$Y_{i+1} - (1+e^{-Q})Y_i + e^{-Q} Y_{i-1} = 0 \quad (Q > 0) \qquad (37a)$$

$$e^{Q} Y_{i+1} - (e^{Q}+1)Y_i + Y_{i-1} = 0 \quad (Q < 0) \qquad (37b)$$

A tridagonal iteration scheme solves Eqs (37) (3:10). To scale the grid, a ratio of δ's at different x locations is set up and like terms cancelled. The relationship between any two x locations becomes

$$Y_2 = Y_1 \, (X_2/X_1)^{1/2} \qquad (38)$$

This study defines Y_1 and X_1 and solves the normal grid equation at the far end of the plate. For each given X_2 location, a Y_2 value results. Fig. 4 shows the initial grid for the optimized solution. To optimize the grid, the Q that minimizes the truncation error in the third derivative of the tangential velocity in the computational plane, $U_{\eta\eta\eta}$, must be found. Therefore, the grid must now be transformed into the computational plane.

Although the grid is computed in the form presented

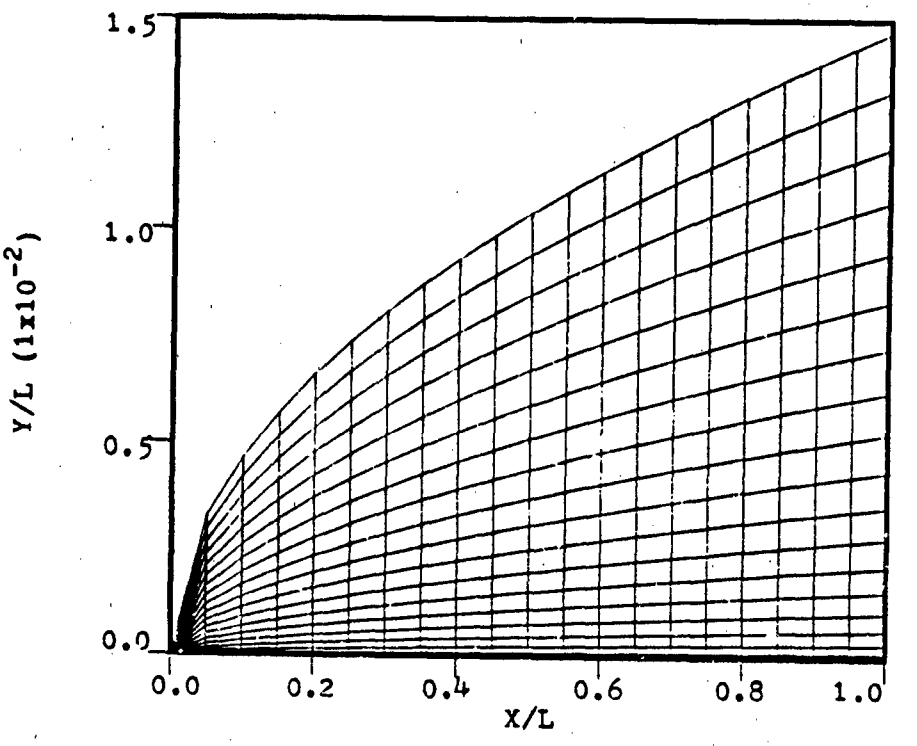


Fig. 3a Power Law Grid

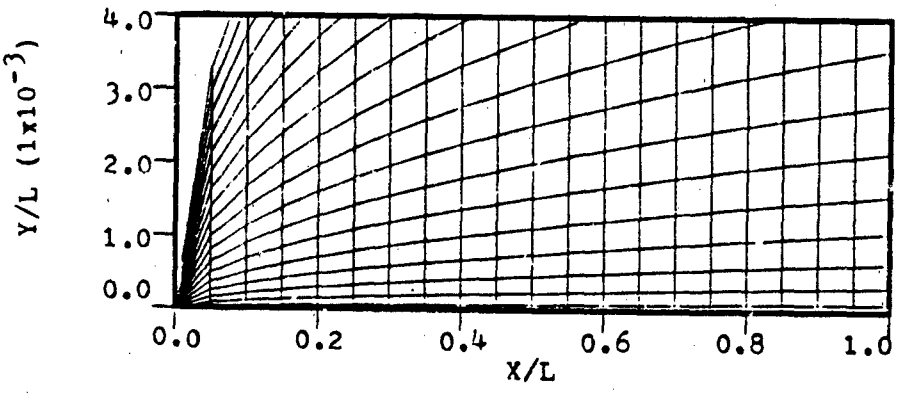


Fig. 3b Expanded Power Law Grid

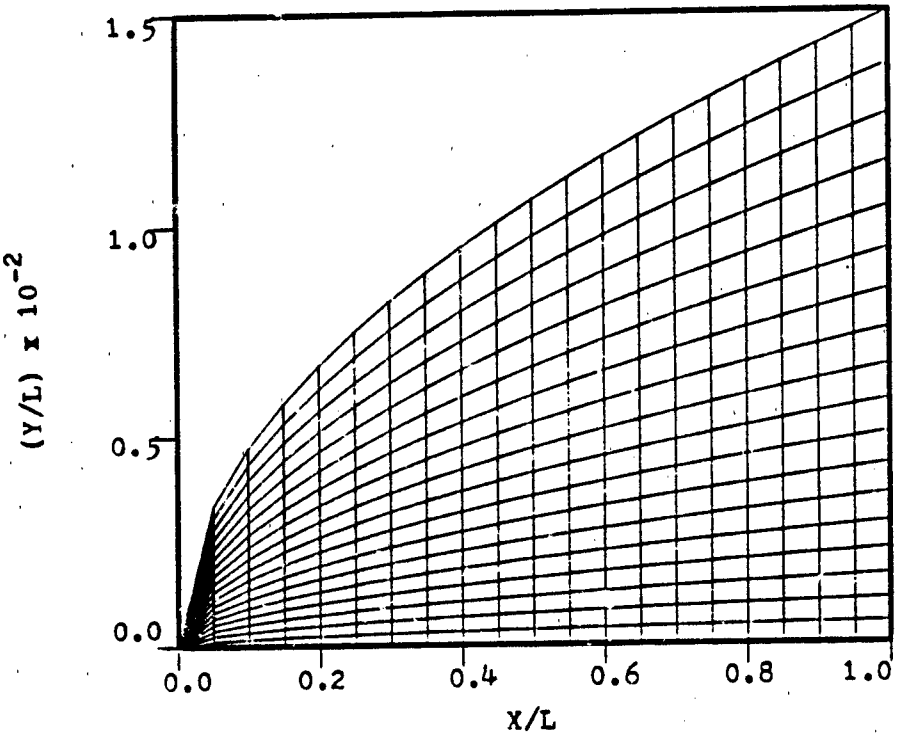


Fig. 4a Exponential Grid

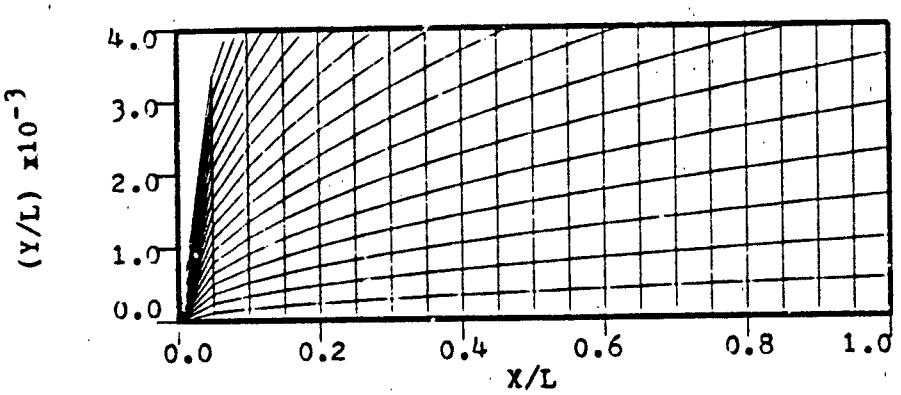


Fig. 4b Expanded Exponential Grid

above, this grid is related to computational plane variables, ξ, η, and τ, by metric coefficients. The functional relationships between the transformed variables and computational variables are

$$\begin{aligned} \xi &= \xi(X,Y,t) \\ \eta &= \eta(X,Y,t) \\ \tau &= t \end{aligned} \tag{39}$$

Using the chain rule and solving for the derivative of each computational variable with respect to the transformed variables, the metric coefficients are

$$\begin{aligned} \xi_X &= Y_\eta/J & \eta_X &= -Y_\xi/J & \tau_X &= 0 \\ \xi_Y &= -X_\eta/J & \eta_Y &= X_\xi/J & \tau_Y &= 0 \\ \xi_t &= (X_\eta Y_t - X_t Y_\eta)/J & \eta_t &= (X_t Y_\xi - X_\xi Y_t)/J \end{aligned} \tag{40}$$

$$J = X_\xi Y_\eta - X_\eta Y_\xi$$

For this study, X does not change in the normal direction, so X_η is zero. Also, for steady-state solutions where the grid is not changing with time, X_t and Y_t are zero. The steady-state metrics are

$$\xi_X = 1/X_\xi \qquad \eta_X = -Y_\xi/X_\xi Y_\eta \qquad \eta Y = 1/Y_\eta \tag{41}$$

If the adaptive grid changes with time, the metrics also include the time dependent metrics

$$\xi_t = -X_t/X_\xi \qquad \eta_t = -Y_t/\eta_Y \tag{42}$$

These metric relationships, Eqs (41) and (42), make it pos-

sible to convert the boundary-layer equations into computational variables. The Dorodnitsyn transformation changes the physical grid shown in Fig. 5 to a transformed grid, shown in Fig. 6. The metrics then mathematically change the transformed grid into the uniform, rectangular grid, shown in Fig. 7. The uniform, computational grid makes solving the boundary-layer equations simpler. It also allows for finding an optimium solution grid which is not possible in the physical plane.

The Numerical Differencing Equations

Whether the equations are incompressible or Dorodnitsyn-transformed, the boundary-layer equations are more easily solved if they are transformed into variables of the computational plane. The constant, normalized mesh size of the computational plane simplifies the differencing equations. Eqs (5)-(7) when changed from X and Y in the transformed plane to ξ and η in the computational plane become

Continuity: $$\xi_X U_\xi + \eta_X U_\eta + \eta_Y V_\eta = 0 \qquad (43)$$

Momentum: $$U_t + U(U_\xi \xi_X + U_\eta \eta_X) + V U_\eta \eta_Y = 1/Re[\rho\mu(U_\eta \eta_Y)]\eta_Y \qquad (44)$$

Energy: $$H_t + U(H_\xi \xi_X + H_\eta \eta_X) + V H_\eta \eta_Y =$$
$$p_t/\rho + \left(\frac{\mu\rho}{PrRe} H_\eta \eta_Y\right)\eta_Y + \left[\frac{\rho\mu(Pr-1)}{2RePr}\, \eta_Y \left(\frac{U^2/2}{}\right)_\eta\right]_\eta \eta_Y \qquad (45)$$

(15:23). For the momentum and energy equations, the differencing methods used are three-point, windward differencing on the first derivative terms, second-order, central differencing on second derivative viscous terms such as $U_{\eta\eta}$, $H_{\eta\eta}$, and $T_{\eta\eta}$, and two-point, backward differencing on the time

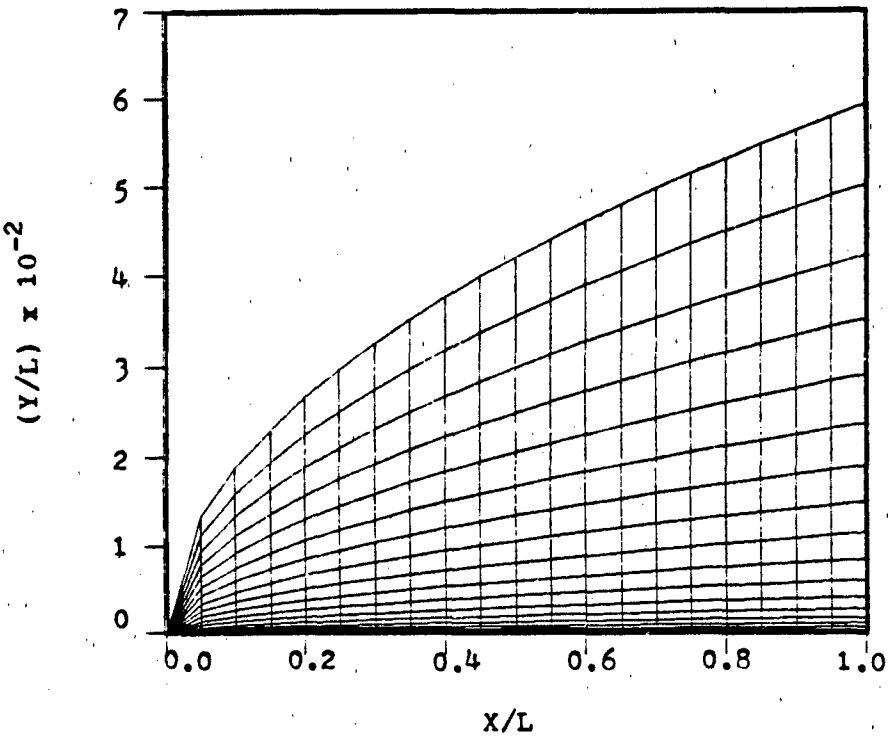


Fig. 5a Physical Grid

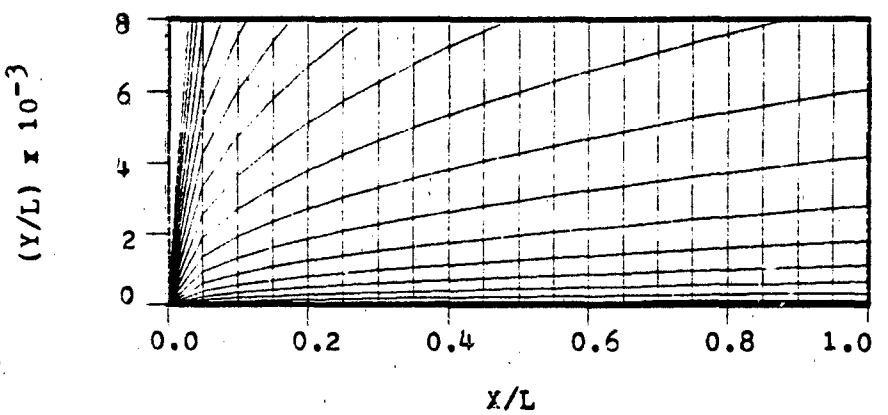


Fig. 5b Expanded Physical Grid

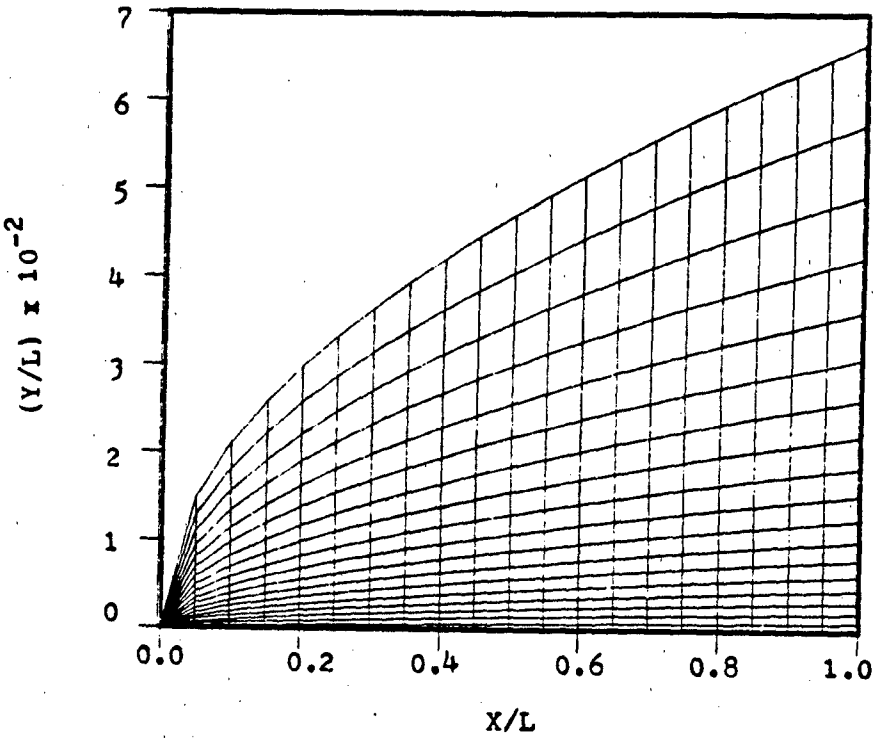


Fig. 6a Transformed Grid

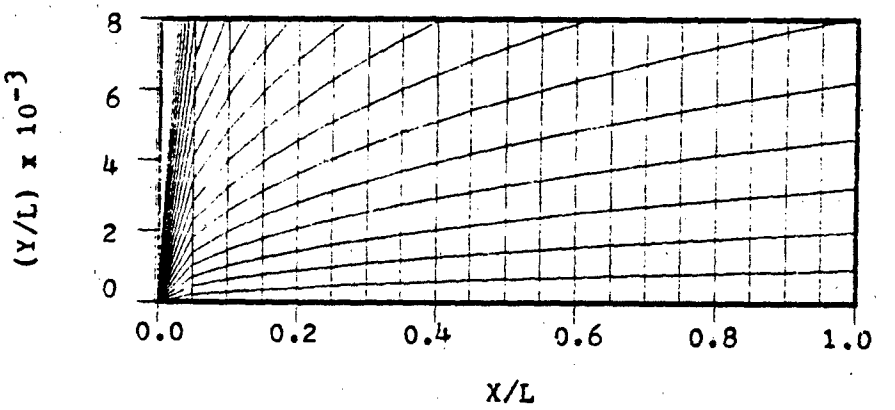


Fig. 6b Expanded Transformed Grid

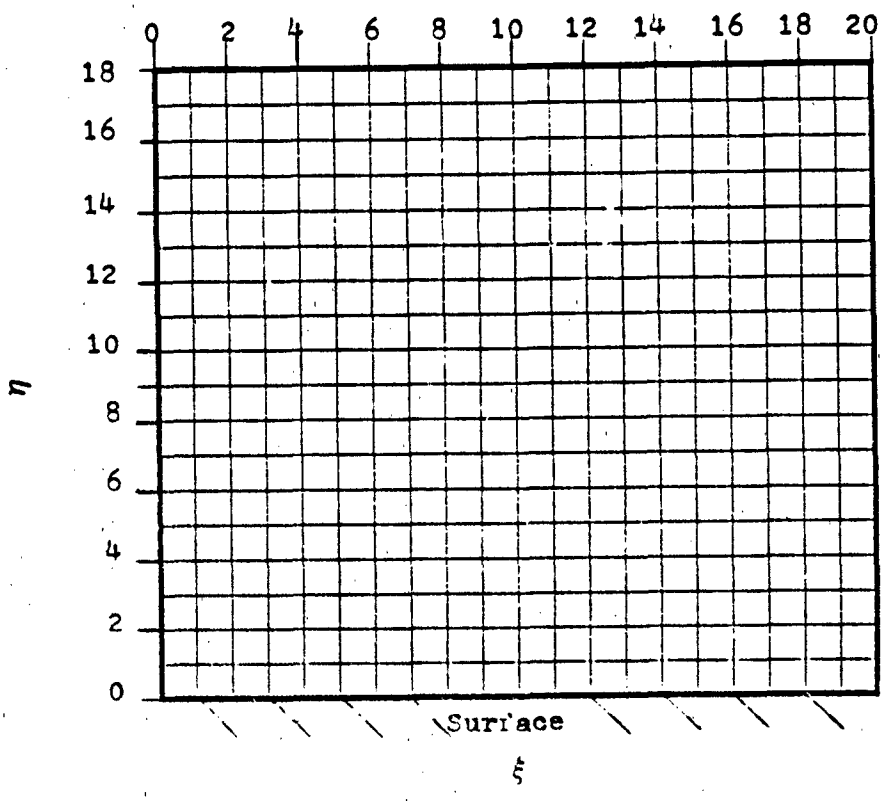


Fig. 7 Computational Grid

terms. At the boundaries, where three points cannot approximate the first derivatives, the differencing is two-point, upwind. This method is first-order accurate in time and second-order accurate in space, except at the boundaries where it is first-order accurate in space. Lange uses this method because, "this differencing scheme guarantees diagonal dominance and is second-order accurate in the computational plane" except at the boundary points as noted above (15:24). After applying the differencing to Eqs (44) and (45), and solving for $U_{i,j}$ and $H_{i,j}$, the solution equations result.

$$\begin{aligned} \text{Momentum:}\quad U^n{}_{i,j} = \{ & U^{n-1}{}_{i,j} + a\xi_X\,\Delta t U U^n{}_{i-1,j} \\ & - b\xi_X\,\Delta t U U^n{}_{i-2,j} + [c\eta_X U\,\Delta t + c\eta_Y V\,\Delta t \\ & + \eta_Y\,\Delta t((\rho\mu)_{j-1/2}\,\eta_Y)/Re]U^n{}_{i,j-1} \\ & - [d\eta_X U\,\Delta t + d\eta_Y V\,\Delta t]U^n{}_{i,j-2} \\ & + \eta_Y\,\Delta t[(\rho\mu)_{j+1/2}\,\eta_Y]U^n{}_{i,j+1}/Re]/U_{coef} \end{aligned} \tag{46}$$

$$\text{where } U_{coef} = 1 + e\xi_X U\,\Delta t + f\eta_X U\,\Delta t + f\eta_Y V\,\Delta t + \eta_Y\,\Delta t\,[(\rho\mu)_{j+1/2}\,\eta_Y + (\rho\mu)_{j-1/2}\,\eta_Y]/Re$$

$$\begin{aligned} \text{Energy:}\quad H^n{}_{i,j} = \{ & H^{n-1}{}_{i,j} + a\xi_X U\,\Delta t H^n{}_{i-1,j} - b\xi_X U\,\Delta t H^n{}_{i-2,j} \\ & + [c\eta_X U\,\Delta t + c\eta_Y V\,\Delta t + \eta_Y \frac{\Delta t}{PrRe}((\rho\mu)_{j-1/2}\,\eta_Y)]\,H^n{}_{i,j-1} \\ & - (d\eta_X U\,\Delta t + d\eta_Y V\,\rho t)H^n{}_{i,j-2} + \frac{\eta_Y}{Pr}\frac{\Delta t}{Re}[(\rho\mu)_{j+1/2}\,Y]H^n{}_{i,j+1} \\ & + \frac{\eta_Y\,\Delta t}{2(1-Pr)Re}\,[(\rho\mu)_{j+1/2}\,Y(U^n{}_{i,j+1})^2 \\ & - ((\rho\mu)_{j+1/2}\,\eta_Y + (\rho\mu)_{j-1/2}\,\eta_Y)(U^n{}_{i,j})^2 \\ & + (\rho\mu)_{j-1/2}\,\eta_Y(U^n{}_{i,j-1})^2]\}/\,H_{coef} \end{aligned} \tag{47}$$

$$H_{coef} = 1 + e\xi_X U^{n-1}{}_{i,j}\,\Delta t + f\eta_X U^{n-1}{}_{i,j}\,\Delta t + f\eta_Y V^{n-1}{}_{i,j}\,\Delta t + \frac{\eta_Y}{RePr}\frac{\Delta t}{}[(\rho\mu)_{j+1/2}\,\eta_Y + (\rho\mu)_{j-1/2}\,\eta_Y]$$

The i and j give the abscissa and ordinate locations, respectively, in the computational plane. In the equations above, the coefficients, a - f, change depending on the grid point. Using two-point backward or forward, or three-point forward or backward differencing changes the coefficients. Table I has the coefficients for each case. To linearize the equations, Lange lags the the first iteration solution in time so U, V, and H are calculated based upon the U, V, and H at the same location but previous time step. For subsequent iterations, U, V, and H are calculated based upon the U, V, and H from the previous iteration step. Next, the metric coefficients need to be differenced.

From the previous discussion of the metrics, the only non-zero metrics are ξ_X, η_X, and η_Y. The inverse relationships from Eq (41) generate the expressions for the differenced metrics. Since ξ_X equals X_ξ and a constant spacing between streamwise points is used, a first-order, central difference for X_ξ determines a constant ξ_X. η_Y is found

TABLE I

COMPUTATIONAL PLANE DIFFERENCE COEFFICIENTS

i	j	a	b	c	d	e	f
2	2	1.0	0.0	1.0	0.0	1.0	1.0
2	3-	1.0	0.0	2.0	0.5	1.0	1.5
2	jmax-1	1.0	0.0	-1.0	0.0	1.0	1.0
3-	3-	2.0	-0.5	2.0	-0.5	1.5	1.5
3-	jmax-1	2.0	-0.5	-1.0	0.0	1.5	-1.0
imax-1	jmax-1	2.0	-0.5	-1.0	0.0	1.5	-1.0

similarly by using Y_η, which is not constant. The spacing in the normal direction does change with each streamwise location. Lange uses three, different schemes depending on the location of Y_η in the equation (15:26). Y_η in the viscous terms is differenced with a second-order, central scheme about the $j+1/2$ or $j-1/2$ points. This gives an overall second-order accurate Y_η term. Y_η for all other terms is represented by a second-order, central difference for the interior points. This study uses a three-point windward scheme at the boundaries. This guarantees second-order accuracy for all Y_η differencing. For Y_ξ, Lange uses an analytic calculation for the metric. Lange explains,

> Initially, Y_ξ was calculated using a central difference with a backward difference at the end of the domain. A large amount of leading edge error was induced by this method (15:27).

This study uses an analytic metric different from Lange's.

$$Y_\xi = \frac{.5Y}{X_\xi - X} \tag{48}$$

However, since an analytic solution over the entire plate limits the method's general applicability, it is preferable to use the analytic solution only at the leading edge. Leading-edge error is inherent in the boundary-layer solution if Y_ξ is numericaly calulated. However, if Y_ξ is calculated analytically with Eq (48) at the first couple of streamwise locations, the numerical solution can then be calculated on the remaining points downsteram of the leading edge with less error. Consequently, Y_ξ is calculated with Eq (48) at the

first two streamwise locations and with numerical, difference equations at the rest of the streamwise locations. The numerical, difference equations use central differencing on the interior and three-point, backward differencing at the back of the plate. This results in Y_ξ with the same second-order accuracy as Y_η and X_ξ. Lange gives an example of the resulting fully-differenced, linearized, momentum and energy equations (15:28). Using Lange's SOR method to compute U and H from the differenced equations gives values used in the continuity solution for V (15:30-32).

The transformed, continuity equation, Eq (43), now contains only one unknown. V has not been solved. Expressing V_η in terms of the known metrics and U, V becomes

$$V_\eta = \frac{-Y_\xi}{X_\xi} U_\xi + \frac{Y_\xi}{X_\xi} U_\eta \tag{49}$$

Lange integrates Eq (49) with the trapezoidal rule for a constant step of one. Finite differencing on U_ξ is a three-point, windward scheme (15:32). The integration of U_η used in this thesis is different than Lange's treatment. The U_η term separates into two parts.

$$\frac{Y_\xi}{X_\xi} U_\eta = \frac{1}{X_\xi} [Y_\xi U - \int U (Y_\xi)_\eta \, d\eta] \Big|_{j-1}^{j} \tag{50}$$

The integral is evaluated using a trapezoidal rule with Y_ξ averaged about the j-1/2 point prior to the integration. After collecting terms, the difference equation for the interior points is

$$V_{i,j} = V_{i,j-1} + \{(Y_{\xi i,j} + Y_{\xi i,j-1})*(U_{i,j} - U_{i,j-1}) + (Y_{i,j} - Y_{i,j-1})*[-0.75(U_{i,j} + U_{i,j-1}) + 1.0(U_{i-1,j} + U_{i-1,j-1}) - 0.25(U_{i-2,j} + U_{i-2,j-1})]\} \quad (51)$$

Solving for V completes the flow solution. All parameters of the boundary layer problem are then known or can be determined from U, V, and H.

To determine how close the computed boundary layer solution duplicates the heat transfer coefficient, h, Eqs (26) and (27) are solved numerically. Since

$$k' = \mu' C_p / Pr \quad (52)$$

the transformed, heat equation becomes

$$q'_o = -\frac{\mu' C_p \rho_w'}{Pr}\left(\frac{dT'}{dY'}\right)_{Y=0} \quad (53)$$

The $dT'/dY')_{Y=0}$ term is non-dimensionalized and Eq (53) is then transformed to the computational space (15:33). This equation is

$$q_o' = -\frac{\mu\mu_\infty U^2 \rho_w}{PrLR_e}\left(\frac{dT}{d\eta}\right)\eta_Y \quad (54)$$

where $dT/d\eta = (-3\ T_{i,1} + 4\ T_{i,2} - T_{i,3})/2$ and

$$\eta_Y = 1/Y_\eta = 2/(-3Y_{i,1} + 4Y_{i,2} - Y_{i,3})$$

The equation differencing is three-point upwind for the metric Y_η and $dT/d\eta$. Eq (26) is rearranged so that

$$h' = q_o'/(T_w' - T_{aw}') \quad (55)$$

(22:95). The heat transfer coefficient calculated is then compared to the theoretical h from Eq (32). If the numerical scheme is accurate the ratio of the two h's should be 1.0 for incompressible cases.

Another parameter that is calculated to compare the computer solution to theoretical incompressible results is C_f. Eq (9) gives the Blasius solution for the skin friction coefficient. This relation is derived from the definition of C_f which is

$$C_f = \frac{2\tau_w}{\rho u'^2_\infty} = \frac{2\mu'}{\rho u'^2_\infty}\left(\frac{\partial U'}{\partial Y'}\right)_{Y=0} \tag{56}$$

The non-dimensionalized, transformed equation is

$$C_f = \left(\frac{2\mu_w \rho_w}{\rho_\infty U_\infty L}\right)\left(\frac{U_\eta}{Y_\eta}\right)_{Y=0} \tag{57}$$

Three-point, backward differencing evaluates U_η and Y_η at the wall. When the C_f computed from Eq (57) is divided by the theoretical C_f from Eq (9), the best solution produces a ratio closest to 1.0 for the incompressible case.

Boundary Conditions

The flow conditions at the wall and the upper boundary of the domain fix the boundary conditions. Assuming no slip conditions at the wall, u_w and U_w are zero. Also, there is no flow through the boundary surface, therefore, v_w and V_w are zero. The surface temperature, T_w, for the isothermal wall cases is 530°R. Freestream temperature is also 530°R

for most of the incompressible runs. These cases come close to an adiabatic, wall condition. At the domain's upper boundary, the edge conditions depend on the given Mach number. For subsonic flows, the edge conditions are the same as the freestream conditions. At Mach 1 and above, oblique shock wave theory sets the flow conditions behind the shock. The conditions behind the shock are the edge conditions for supersonic flows. Unlike these fixed boundary conditions, the initial conditions change.

Initial Conditions

For the first iteration of the numerical solution, an initial approximation of U, H, and V must be assumed. A cubic, velocity profile is a relatively simple approximation to the boundary layer and, as shown in Schlichting, it comes close to Blasius', exact, boundary-layer, velocity profile (23:206). The cubic, velocity profile is

$$u = u_e'[1.5\ (y'/\delta) - .5\ (y'/\delta)^3]/u_\infty \qquad (58)$$

For compressible solutions, Y' replaces y' and $\Delta(X)$ replaces $\delta(x)$ in Eq (58). The enthalpy profile is derived from a temperature profile which is an approximate solution of the energy equation (22:34).

$$\frac{T - T_w}{T_e - T_w} = 1.5\ \frac{(y')}{(\delta_t)} - .5\left[\frac{(y')}{(\delta_t)}\right]^3 \qquad (59)$$

Since the total enthalpy, H, is $C_pT+U^2/2$ for isentropic flows, Eq (55) yields the non-dimensionalized guess for H.

$$H_{i,j} = H_w + (H_e - H_w)\left[1.5\ \frac{y}{\delta_t} - .5\left(\frac{y}{\delta_t}\right)^3\right] + .5\left(\frac{U_{i,j}}{U_\infty}\right)^2 - .5\left(\frac{U_e}{U_\infty}\right)^2\left[1.5\ \frac{y}{\delta_t} - .5\left(\frac{y}{\delta_t}\right)^3\right] \quad (60)$$

The thermal, boundary-layer thickness, δ_t, in Eqs (59) and (60) is

$$\delta_t = \delta/(1.026\ Pr^{1/3}) \quad (61)$$

This value for δ_t is valid if the entire plate surface is heated and if δ_t is zero at the front of the plate (22:35). For Pr=1, δ_t will be less than δ. Finally, initial v values are assumed to be zero throughout the domain. These complete the initial guess for the flow solution. The guess is input along with the grid, and solution of the difference equations gives a final, numerical solution for the flow characteristics. Unfortunately, the input grid may not be able to adequately resolve some of the gradients in the solution. Therefore, an optimized grid which minimizes the truncation error and produces better flow solutions must be found.

Chapter IV: Grid Optimization

Using finite-difference methods to solve the boundary-layer equations requires polynomial approximation of the first derivatives of velocity and enthalpy. The polynomial approximations are not exact. Truncation error is inherent in the solution since all terms of the approximation are not included in the calculations. In the case of second-order differencing of velocity derivatives, truncation error is some multiple of the third derivative terms such as $U_{\eta\eta\eta}$ and $U_{\xi\xi\xi}$. Minimizing the third derivative terms insures the truncation error is small, and the velocity solution is consequently most accurate. Since the velocity gradients in the streamwise direction are small compared to the velocity gradients in the normal direction, this study minimizes only $U_{\eta\eta\eta}$ terms. Also, as $U_{\eta\eta\eta}$ approaches zero, $U_{\eta\eta}$ approaches a constant and $U_{\xi\xi}$ approaches zero. Finding the solution grid which minimizes $U_{\eta\eta\eta}$ thus optimizes the boundary-layer solution. Powell's, conjugate-direction method which iterates on an n-dimensional array until finding the minimum of a desired function without calculating the derivatives of the function, is well-suited to this problem. Powell's method finds the grid control function, Q, in Eq (35) which forms the optimum grid.

Truncation error is present in any finite-difference solution which uses polynomial approximations to define gra-

dient terms. The finite differencing of the boundary-layer equations, uses Taylor, polynomial expansions of the gradient terms. For example, the first-order, three-point, backward difference of a gradient term, W_η is

$$W_\eta = \frac{3}{\quad}_{i,j} \frac{-4\ W}{\quad}_{i,j-\frac{1}{2}} \frac{+\ W}{\quad}_{i,j-2} + \frac{1}{3} W_{\eta\eta\eta} + \frac{1}{4} W_{\eta\eta\eta\eta} + \ldots \quad (62)$$

assuming $\Delta\eta$ is 1. Since the first three terms replace a gradient term, like U_η, in the finite-differenced, boundary-layer equations, the truncation error is the sum of all the higher derivative terms. The most heavily weighted term of the truncation error is the $W_{\eta\eta\eta}$ term. If $W_{\eta\eta\eta}$ is zero, $W_{\eta\eta\eta\eta}$ is zero. Therefore, minimizing the third derivative terms should significantly decrease the truncation error in the finite-difference solution.

The boundary-layer equations are two-dimensional equations. For a completely, accurate solution, two-dimensional, grid equations should define the solution grid. However, this study uses the characteristics of the incompressible, boundary layer to simplify the grid selection. One of the boundary-layer simplifications of the Navier-Stokes equations is that the normal velocity gradients are much greater than the streamwise, velocity gradients. Therefore, any error in the normal velocity gradients has a larger impact on the overall error of the computer solution than error in the streamwise gradients. Making the x locations constant in the adaptive grid allows the grid to concentrate on optimiz-

ing only the gradients in the normal direction. Primarily, the adaptive grid tries to minimize the sum of $U_{\eta\eta\eta}{}^2$, but it can also minimize the sums of $U_{\eta\eta}{}^2$, $U_{\eta}{}^2$, or some other combination of these terms. Once the normal direction is optimized, the streamwise dimension is scaled according to Blasius' description of the incompressible, boundary layer. A parabolized, grid scheme can be used to avoid this scaling. Eq (38) shows a square-root dependence on each streamwise coordinate of the Blasius solution. Ordinate locations found using a given grid control function and the normal grid equation, Eq (35), at one x location are scaled on the rest of the grid using Eq (38). Powell's minimization method finds the optimum, grid control function, Q, which produces an optimized flow solution.

Powell's Method

Powell's, minimization method has several features which make it attractive for optimizing the normal gradients. Powell's method does not require calculating derivatives of the function being minimized. This is especially helpful in this study where no explicit function' is being minimized. Powell's method is flexible. Any computed quantity can be minimized. Any parameter can become the minimized function in Powell's algorithm. This allows the optimization of the grid with several, different, minimized parameters to find the best optimization technique. Powell's method assumes a locally, quadratic form between points rather than a linear

form. Powell's method also converges quickly in a finite number of steps. Powell explains,

> when the procedure is applied to a quadratic form ...the ultimate rate of convergence is fast when the method is used to minimize a general function (19:155).

Powell's use of two conjugate directions to search for the minimum achieves this fast convergence. Therefore, Powell's method is accurate, allows fast convergence, is flexible, and does not require further derivatives of $U_{\eta\eta\eta}$ to minimize this parameter.

This study adapts Powell's method to minimize the sum of $U_{\eta\eta\eta}^2$, $U_{\eta\eta}^2$, U_{η}^2, or any combination of these sums over the entire solution grid. One iteration of Powell's minimization procecure is explained below.

> (i) For r=1,2,...,n calculate λ_r so that $f(A_{r-1} + \lambda_r E_r)$ is a minimum and define $A_r = A_{r-1} + \lambda_r E_r$.
>
> (ii) Find the integer m, $1 \leq m \leq n$, so that $\{f(A_{m-1})-f(A_m)\}$ is a maximum, and define $\Delta=f(A_{m-1})-f(A_m)$.
>
> (iii) Calculate $f_n= f(2A_n-A_0)$, and define $f_1= f(A_0)$ and $f_2= f(A_n)$.
>
> (iv) If either $f_3 \geq f_1$ and/or $(f_1 - 2f_2 +f_3)*(f_1-f_2-\Delta)^2 \geq .5\Delta(f_1-f_3)^2$ use the old directions $E_1, E_2,..., E_n$ for the next iteration and use A_n for the next A_0, otherwise
>
> (v) defining $E= (A_n-A_)$, calculate λ so that $f(A_n + \lambda E)$ is a minimum, use $E_1, E_2,...,E_{m-1}, E_m, E_{m+1}, E_{m+2},...E_n, E$ as the directions and $A_n+\lambda E$ as the starting point for the next iteration (19:156).

For this study, the sum of $U_{\eta\eta\eta}^2$, $U_{\eta\eta}^2$, U_{η}^2, or a combination of any of these terms is the above function, f. The A_0 to A_n parameters above are $A(_1)$, A(2),..., A(n). Q, the con-

trol function, is some function of the A(1) to A(n) parameters.

$$Q = Q\,(A(1),\ A(2),\ldots,A(n)) \tag{63}$$

The λ directions that Powell mentions are the directions the algorithm searches to find the function's minimum. Initially, this study uses two parameters so that

$$Q = A(1) + A(2) \tag{64}$$

The first step in the minimization procedure is finding $f(A_0)$ which is also f_1. The flow chart in Appendix C shows how program MNTRER in Appendix D applies Powell's algorithm.

The program MNTRER in Appendix D finds each λ_r by iterating over k iterations until it reaches a minimum of $f(A_{r-1} + \lambda_r E_r)$. For each r, λ is initially zero. Two other values of λ, λ^+ and λ^-, are also chosen a fixed percentage distance from λ. The k iteration then begins. Three new parameters, A^+, A^-, and A^b, are calculated for each r with λ^+, λ^-, and λ.

$$A_r(\ ) = A_r(\ ) + \lambda_r E_r \tag{65}$$

The new A values add according to Eq (64) and form Q^+, Q^-, and Q^b. Three new grids are then generated for the +, -, and b cases. Three sets of boundary-layer equations are solved. Taking the three boundary-layer solutions, subroutine NEWP1 calculates U_η, $U_{\eta\eta}$, and $U_{\eta\eta\eta}$ and three functional values of f, f^+, and f^-. Subroutine NEW[illegible] also fits a quadratic to

λ^+, λ^-, and λ. It then finds the minimum of the quadratic to produce a new λ. Eq (65) then finds a new A^N. From this a new Q is found, and the solution grid is generated. Then, the boundary-layer equations are solved, and f_N is determined. The minimum of f^+, f^-, f, and f_N is determined. The λ associated with the minimum becomes the new λ for the next k iteration. A new value of A_r is also found with the new λ. Then, the k iteration increments and continues until reaching preset maximum k or the difference between the new and old λ is sufficiently small.

After the k iteration, Powell's step (i) is complete. The program then continues with the other steps. Step (ii) compares the f values for each final A_r and determines the r which maximizes $f(Ar_{-1} - A_r)$. Program MNTRER then finds f_3 in Powell's step (iii). The program then performs the tests in step (iv). If the tests are true, the E_r values of the first iteration remain the same and another iteration is run. If the tests are false, step (v) calculates a new value of E equal to $A_n - A_0$. Using the same steps as the previous k iteration, the program finds the λ which minimizes $f(A_n + \lambda E)$. $A_n + \lambda E$ then replaces A_0 as the starting point for the next full iteration of the procedure. The new E calculated in step (v) also replaces the E_r in which the largest decrease was made in the last iteration. The whole procedure is repeated until the changes between A_r's are sufficiently small or the maximum number of iterations is reached. Powell's optimization is then complete.

Powell's optimization method, as adapted in program MNTRER, provides some flexibility to test which minimized function gives the most accurate results. Since Powell's method minimizes the function, f, any function f can be chosen. This study checks the affect of minimizing other derivatives, or a combination of derivatives, on the accuracy of the final, optimized, boundary-layer solution. The sum of U_{η}^{2}, $U_{\eta\eta}^{2}$, $U_{\eta\eta\eta}^{2}$, or any combination of these three can be the minimized function, f. These are the three functions this study investigates, but any, definable parameter can be minimized by the optimization method. Based on the truncation error argument, minimizing $U_{\eta\eta\eta}$ with Powell's method should produce the most, accurate, finite-differenced, boundary-layer solution.

Chapter V: Results and Discussion

This study shows how an optimized, adaptive-grid solution of a boundary-layer code produces a better description of flow characteristics in the boundary layer. Before looking at the optimization of the boundary-layer code, the characteristics, capabilities, and limitations of the boundary-layer code must be understood. For example, questions which must be answered are how do initial conditions affect the solution, or how many iteration and time steps are necessary to get a converged solution. The boundary-layer program used to solve the finite-differenced boundary-layer equations has several improvement's to Lange's boundary-layer code. Comparison of the computed, boundary-layer solution and Blasius', theoretical solution for incompressible flow over a flat plate shows the increased accuracy of the new, boundary-layer code. Powell's optimization method is then applied to the new, boundary-layer code to optimize the adaptive grid. The optimized, adaptive grid is more accurate than the non-adaptive grid, incompressible solution. The effect of different parameters on the optimization method is investigated. These investigations of the incompressible cases verify the new, boundary-layer code and the optimization method. Then, the optimized, adaptive grid is applied to supersonic and hypersonic, compressible, flow problems. Solutions for compressible flow over the flat plate and wedge

are compared to previous theoretical development. This comparison shows the applicability of the optimized, adaptive grid and the boundary-layer code to compressible, flow problems. The first step however is to verify that the boundary-layer code can solve for the correct, incompressible results.

Incompressible Flow

The first step in determining the applicability of the optimization method verifies that the code reproduces the exact, theoretical results for incompressible flow over a flat plate. The boundary-layer code is set up to handle compressible, flow problems where density and viscosity are changing. The input flow conditions need to model incompressible flow where density and viscosity should be essentially constant. To do this, the incompressible cases are run at Mach .01. Freestream temperature and pressure are set at 530°R and 2116.2 psf. Surface temperature is also a constant 530°R. For these conditions density and viscosity are essentially constant. Throughout the runs, the non-dimensionalized density ranges from .99941217 to .99941725 in the normal, flow direction. The density computed at the edge boundary is .99941725. The non-dimensionalized density should be 1.0 for the entire domain and especially at the edge boundary. The computed ρ is 5.8×10^{-4} % in error. This is insignificant, but the error is present. Lack of more significant figures in the gas constant used and round-off error in the computer cause this error. The computed den-

sity changes less than 5 x 10^{-4}% over the domain. The density is therefore essentially constant. The viscosity ranges from 3.7989885 x 10^{-7} to 3.7989486 x 10^{-7}, the freestream value. With this error of 1 x 10^{-3}%, viscosity is also essentially constant. The input freestream conditions cause the boundary-layer code to calculate essentially an incompressible problem. To compare various characteristics of the boundary-layer code, a 61x30 solution grid is used, except as noted.

Boundary Layer Code.

Convergence Parameters.

Several, input parameters control whether the boundary-layer solution converges on a final solution and how fast it converges on that solution. These parameters are the number of time iterations (NT), the number of iterations at each time step (KT), the size of the time step (DT), the convergence criteria (EPS), and the initial conditions for the solution. Subroutine BLIMP, listed in Appendix D, represents the variables as NT,KT,DT, and EPS, respectively. For a particular time step, the solution is converged when the maximum of the differences of U, V, or H at the old, iteration level and the new, iteration level is below a specified error value. Picking a smaller minimum error tolerance, EPS, results in a more precise answer. However, it also means the solution will need more iterations and computer time to converge. For this study, EPS is 1.0 x 10^{-6}. This gives acceptable precision without unnecessary use of computer time.

Table IIA outlines several different runs of the boundary-layer code using various input parameters.

TABLE IIA

Non-Adaptive Incompressible Grid Solutions- Inputs

Run	NT	KT	IMET*	DT	Pr
1	1	4	3	10^6,	.72
2	5	4	3	10^6,	.72
3	10	4	3	10^6,	.72
4	15	4	3	10^6,	.72
5	20	4	3	10^6,	.72
6	5	10	3	10^6,	.72
7	10	10	3	10^6,	.72
8	15	10	3	10^6,	.72
9	20	10	3	10^6,	.72
10	5	4	3	10^6,	.72
11	10	4	3	10^6,	.72
12	15	4	3	10^6,	.72
13	20	4	3	10^6,	.72
14	1	4	3	10^6	.72
15	20	4	3	10^6,	.72
16	5	4	3	10^2	.72
17	10	4	3	10^2	.72
18	15	4	3	10^2	.72
19	20	4	3	10^2	.72

Note: All solutions use a 61x30 Power Law solution grid.
* - number of analytic points in the solution

The ratio, Cf/Cf_t, listed in Table IIB shows how close the computed solution comes to Blasius', theoretical solution.

TABLE IIb

Non-Adaptive Incompressible Grid Solutions- Results

Run	Cf/Cf_t (leading edge)	Cf/Cf_t (trailing edge)	DELT* ($\times 10^{-4}$)	DELK* ($\times 10^{-4}$)
1	.91378	.90550	4,840	4.1
2	.97052	.97400	5.5	.117
3	.98740	.98961	6.1	.185
4	.99203	.99193	.916	.0277
5	.99328	.99229	.168	.00592
6	.98827	.99231	7.84	.00996
7	.99332	.99239	.742	.00625
8	.99370	.99241	.0505	.00433
9	.99373	.99241	.00199	.00199
10	.97052	.97398	55.3	1.17
11	.98740	.98971	6.1	.186
12	.99203	.99193	.918	.027
13	.99328	.99229	.168	.00592
14	.91378	.90550	342.0	8.99
15	.99328	.99228	.119	.000688
16	.97051	.97374	4821.8	40.9
17	.98740	.98956	55.67	1.178
18	.99203	.99193	6.18	.1887
19	.99328	.99229	.1685	.00060

* : DELK= maximum difference in U, V, or H between iteration steps: DELT= maximum difference in U, V, or H between time steps

The numbers shown are values of Cf/Cf_t at the x locations one point inside the leading edge and at the trailing edge. Figs. 8 - 12 show the trend for how Cf/Cf_t changes in the streamwise direction. Flow properties at the first two points inside the leading edge are calculated analytically and are not changed by the numerical solution. These analytical points are theoretically the closest the solution can come to the flow properties. Leading edge error distorts the results after the analytically calculated points. However, as the solution varies in the streamwise direction, the numerically calculated solution recovers to its true value toward the back of the plate. Therefore, the first point shows the analytic value of the ratios, and the trailing, edge value shows the best, numerical solution for the run. Two other results shown are DELT, the maximum difference of U,V, or H between time steps, and DELK, the maximum difference for U, V, or H between iteration steps. DELT shows the improvement in the solution between successive time steps. If the maximum difference for U, V, or H between iterations steps, DELK, is below $1x10^{-6}$, the solution is converged. The comparison of values in Table II shows the effect of the input parameters on the boundary-layer solution.

A parameter chosen to reduce computer time is the size of the time step, DT. For each iteration of time, the time in the solution increases by DT. Since the boundary-layer equations are solved implicitly by SOR, large, time steps do

not cause any instability in the solution of the boundary-layer equations. The problem will still converge on to the steady-state answer, regardless of the size of the time step. Therefore, to get a converged solution in as few time steps as possible, this study uses a large, non-dimensionalized, time step of 1.0×10^6. In runs 16-20, the solution still converges in 20 time steps using DT equal to 100. Larger DT produces better convergence, however. DELK and DELT go down as higher DT's are used. DELT and DELK in runs 1-5 is smaller than the same two parameters in runs 10-13, and runs 10-13 are smaller than runs 16-19. Other than this improvement in DELT and DELK, increasing DT does not change the solution much. Therefore, to be guaranteed the best, converged solution possible the larger DT is used in the steady-state solutions.

The number of iterations per time step, KT, determines how accurate the results are at each time step. If the number of iterations per time step is large, i.e. 10 in runs 6-9, the program does more iterations of the equations per time step. The question arises whether it is better to have more iterations per time step and fewer time steps for convergence or fewer iterations and more time steps. Runs 6-9 set KT at 10. Runs 1-5 set KT at 4. As expected, convergence at each time level listed is better than for corresponding time steps of runs 1-5. With KT at 10, better results for Cf/Cf_t are obtained at 15 time steps (run 8) than are obtained after 20 time steps with KT of 4 (run 5).

However, 15 time steps with KT at 10 takes 37.5 CPU seconds on the CYBER 6070. For KT at 4, 20 time steps takes 30 CPU seconds. This indicates there are more iterations of the solution with the larger KT. More iterations produce better results for Cf/Cf_t, DELK, and DELT. Since a large DT ensures the solution is converged, the total number of iterations is the important factor. Whether the iterations are mostly with each time step or with more time steps, the converged solution occurs when enough total iterations have been made. Therefore, since acceptable convergence is achieved with less computer time for smaller KT values in runs 1-5, the inputs for run 5 are used as a set of baseline inputs for the rest of the study.

Program Logic Choices.

In this study, three major changes have been made to Lange's, boundary-layer code. The initial conditions are slightly different, the integration of the continuity equation is improved, and some of the metrics are solved more precisely. These changes help the boundary-layer program to more closely conform to boundary-layer theory. The improvement is seen in each case by comparing the ratio of the computed and theoretical values for C_f. The closer this ratio is to 1.0 indicates better accuracy. Therefore, each of the changes is compared to find the best method to get closest to Blasius', exact, incompressible, boundary-layer solution.

The choice of initial conditions depends on the physics

of the flow problem. However, the finite-difference solution converges on the same answer regardless of the initial conditions chosen. The boundary-layer code changes Lange's, initial guess for the enthalpy profile, the thermal, boundary-layer thickness, and the U and H values given to the jmax+1 points. Although the points outside the jmax location are not of interest to the solution, the finite-difference scheme requires that points at jmax+1 be defined. In Lange's program, these jmax+1 values of U and H are set at 0.0. Since these points are on the boundary, well outside the boundary layer, the jmax+1 points should equal the jmax values. These values are equal to the edge conditions. Therefore, at jmax and jmax+1 positions, U and H are defined as U_e and H_e. Lange also assumes the thermal, boundary-layer thickness, δ_t, equals the boundary-layer thickness, δ. This approximation is acceptable, but it is more accurate in the hypersonic limit and for flows with Pr equal to 1. However, for the incompressible case, Eq (59) is accurate in the more, general case where Pr is non-unity and the surface is uniformly heated along its entire length. This study uses this equation to also form Eq (58), a better guess of the enthalpy profile. Lange's guess does not include the last two terms of Eq (58). These three changes to the initial conditions do not affect the final solution. Boundary-layer solutions with all other inputs constant but different, initial conditions converge to the same final solution. Runs 1-5 and 14-15 compare the results of the two runs. The new, initial condi-

tions decrease the accuracy of the solution at the first time step. But the final solution is slightly more accurate with the new initial conditions. Although it is not necessary to know the physics of the problem fully to get the correct initial conditions, accurate initial conditions increase the accuracy of the boundary-layer solution by decreasing the iterations need for convergence.

Another change to Lange's program is in the integration of the continuity equation to solve for V. Eq (51) shows the integration of V_η. Fig. 8 shows the dramatic increase in the accuracy of the new, boundary-layer code. Both runs are converged. The solution shown using Lange's integration method for V_η uses a two-point, windward difference at the boundaries with the two leading-edge points defined analytically. The new, boundary-layer code uses three-point, windward difference with three leading edge points defined analytically. This accounts for the large overshoot for Lange's solution, while the new code undershoots slightly. The important result, seen in Fig. 8, is the new code recovers to more, accurate values of Cf/Cf_t. Lange's code approaches .93319, while the new code approaches .99229. The errors are 6.7% and .71%, respectively. The order of magnitude increase in accuracy for Cf/Cf_t indicates he new, boundary-layer code is better at predicting the incompressible solution.

As mentioned earlier, the new, boundary-layer code uses a different, finite-difference approximation of the metrics, Y_ξ and X_ξ. Using the three-point windward scheme at the

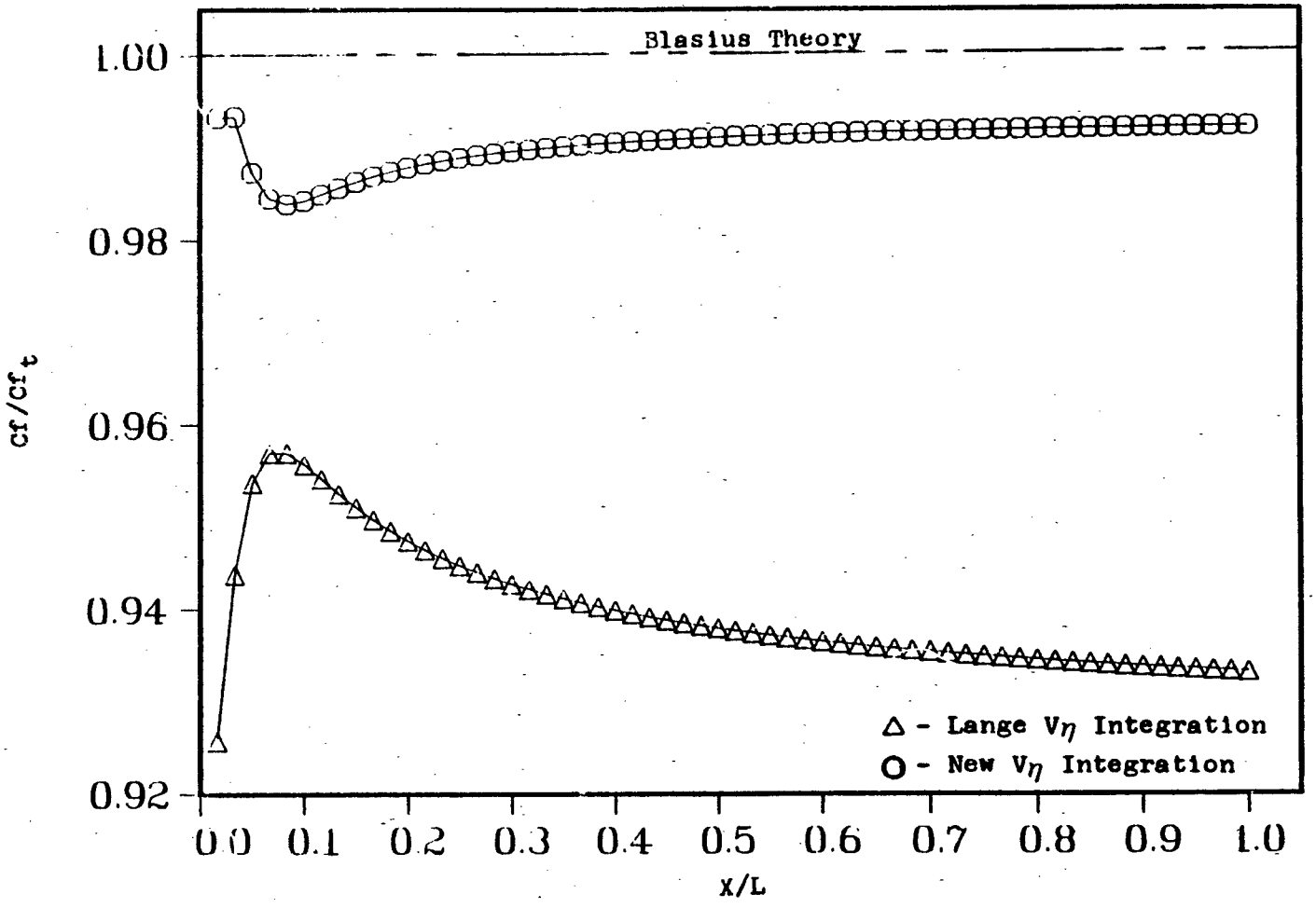


Fig. 8 V_η Integration Method Comparison, Cf/Cf_t

boundaries gives second-order accuracy throughout the metrics solution. One differencing method tried uses two-point, windward differencing which is only first-order accurate. For the X_ξ metric, the differencing does not change the solution. Since the streamwise grid spacing is constant, either differencing scheme gives the same values of X_ξ. The differencing on Y_ξ does matter, though. Using the new integration of V_η, solutions with three-point and two-point, windward differencing on Y_ξ are run. Fig. 9 shows the ratio for C_f/C_{ft} at each streamwise location for each solution. With the two-point, windward scheme leading edge error influences the solution far downstream. The three-point, windward scheme damps the error out more effectively, converging on better values. The three-point, windward scheme is therefore more accurate. Part of the reason for reduced leading edge error in the three-point, windward scheme is this scheme uses three analytic points at the leading edge. The two-point, differencing method only uses two analytic points at the leading edge. In the analytic solution, the Y_ξ metric is known exactly. It is calculated by knowing the exact form of the boundary-layer shape. Eq (48) computes the Y_ξ instead of calculating it from numerical differencing. This takes the numerical error out of the Y_ξ metric. With a completely analytic Y_ξ metric and density and viscosity forced to remain constant, the best C_f ratio obtained with a 61x30 grid is .99361. Figs. 10 compares this completely analytical case to the case with Y_ξ computed numerically except at the first

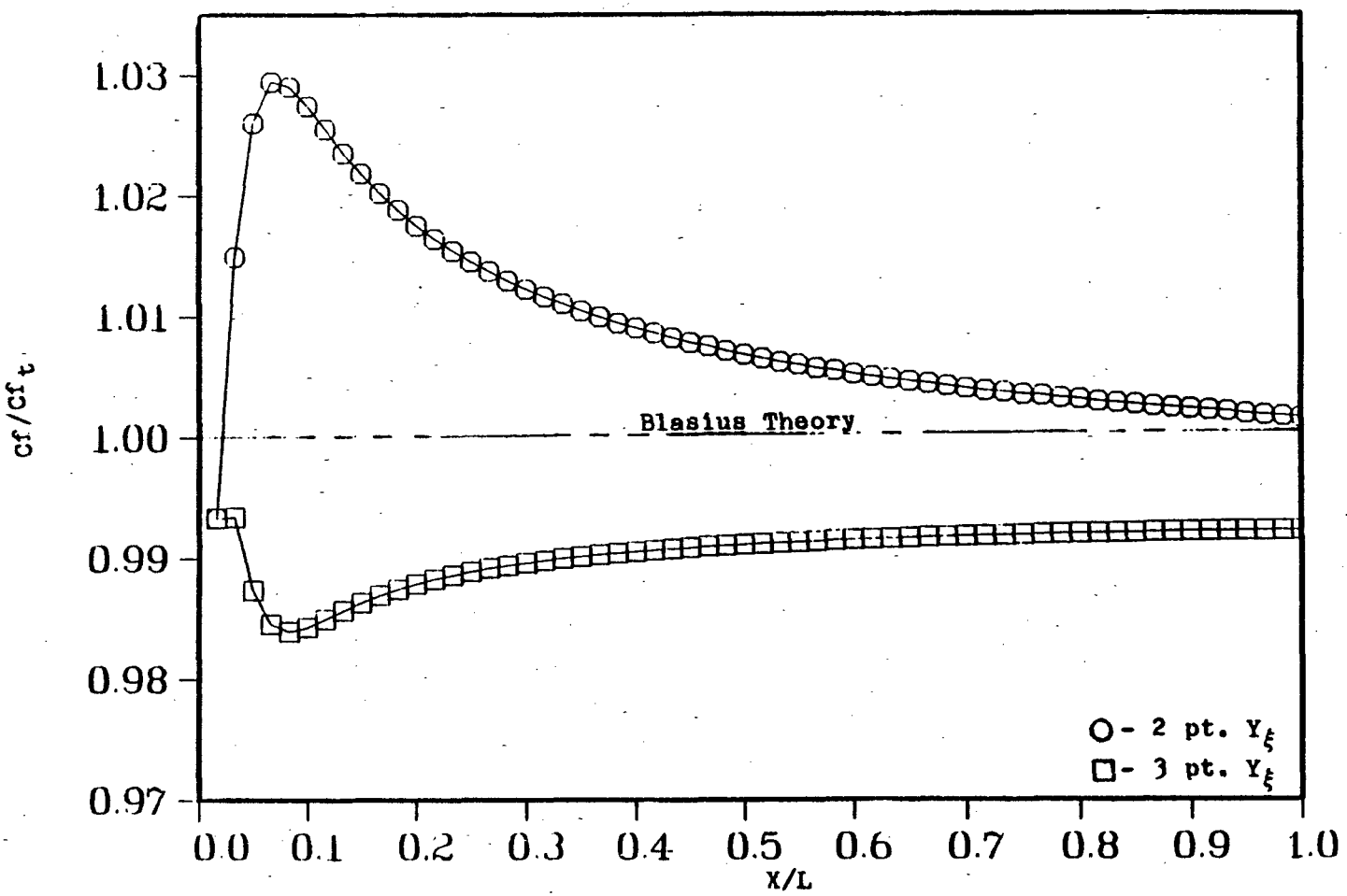


Fig. 9 Y_ξ Differencing Comparison, Cf/Cf_t

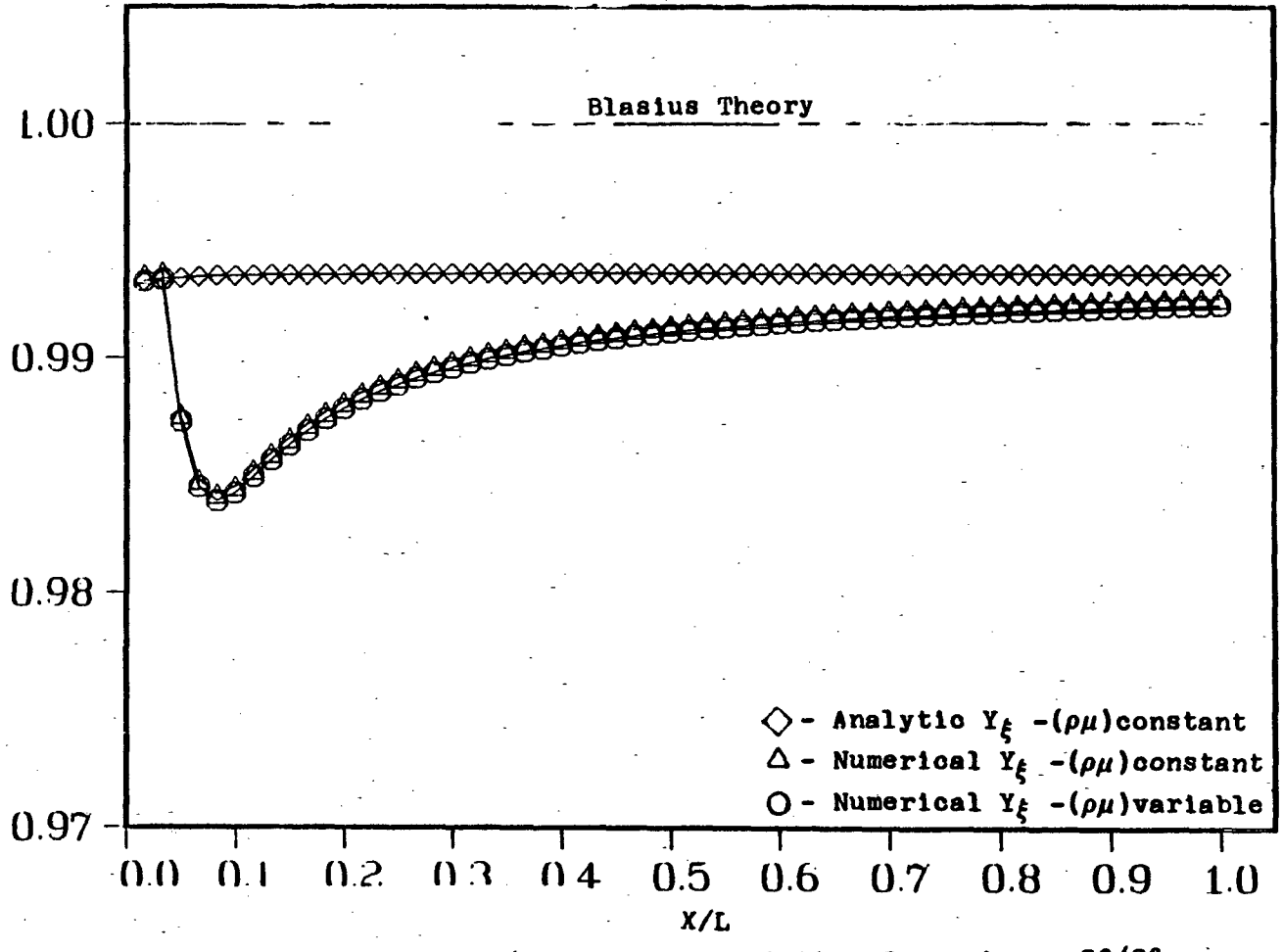


Fig. 10 Analytical/Numerical Y_ξ Solution Comparison, Cf/Cf_t

three, streamwise locations. Density and viscosity are held constant in both runs. The completely analytic case is definitely more accurate for predicting flow in the boundary layer. However, if the analytic metric is used, the boundary-layer code could only be used with a solution grid which has been scaled according to a square-root of X function as in Eq (38). This scaled grid is accurate for similar boundary layers. Since the boundary-layer code needs to be applied to more general problems, the numerical metric is used. Also plotted in Fig. 10 is the solution with the numerical Y_ξ and changing density and viscosity. There is a very slight error between the two numerical Y_ξ solutions caused by allowing density and viscosity to vary. This error does not affect the results, significantly. The new method of determining V_η and three-point differencing on Y_ξ significantly improves the solutions computed by the boundary-layer code. The grid dependence of the finite-difference solution also cannot be ignored.

Grid Dependence.

The number of points in the solution grid and the way they are generated affect the boundary-layer solution. As the number of points in the solution grid increases, the solution's accuracy increases, and the computer time, or iterations, required for a converged solution also increases. Fig. 11 shows the solutions for three, different, size grids. By increasing the number of grid points Cf/Cf_t ap-

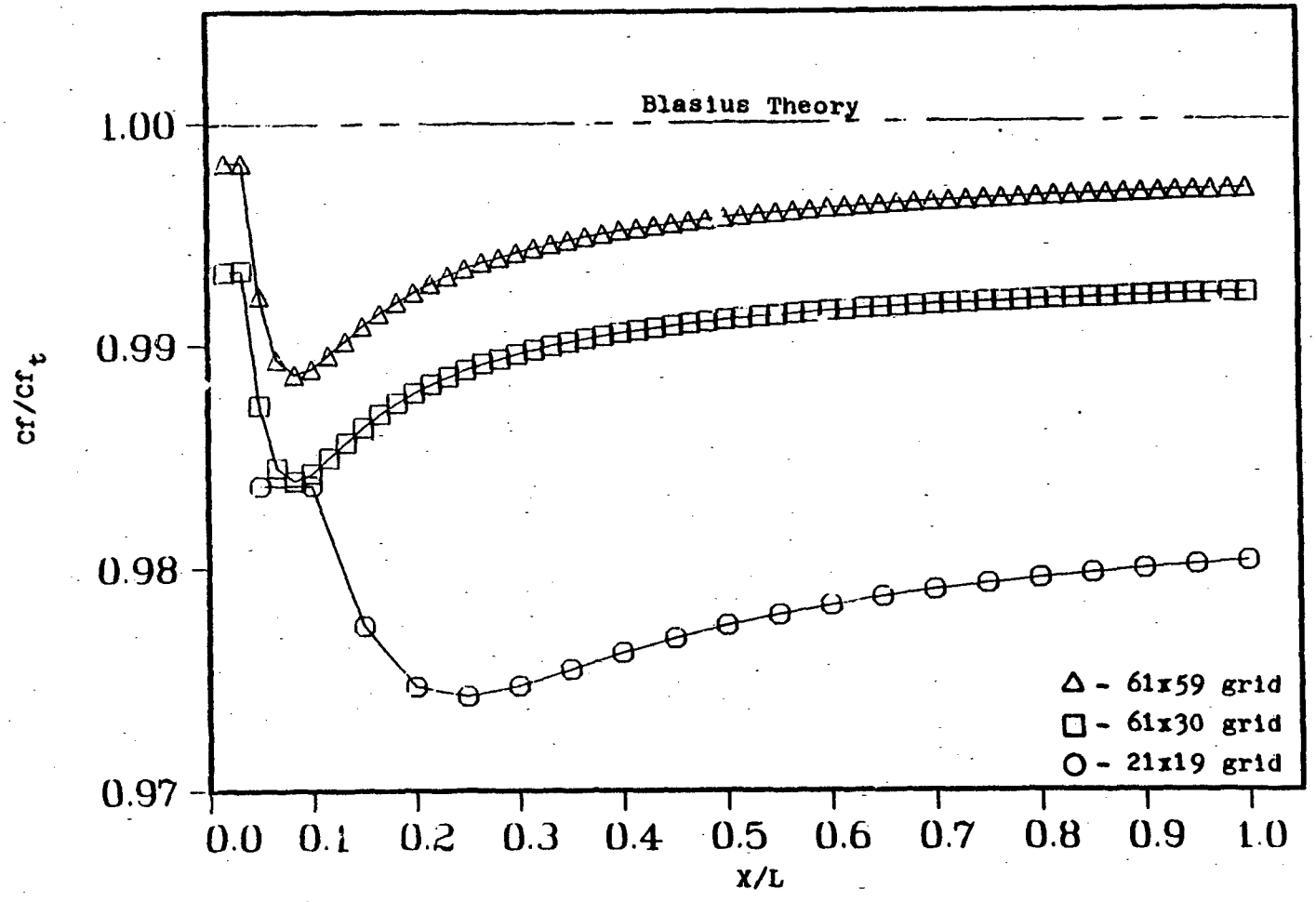


Fig. 11 Power-Law, Grid Size Comparison, Cf/Cf_t

proaches one. However, the 61x59 grid requires 75 time steps and 206.78 CPU seconds to converge on the solution. The 61x30 grid takes 20 time steps and 29 CPU seconds. The 21x19 grid takes 5.97 CPU seconds for 20 time steps. Increasing the number of grid points does increase accuracy but with a considerable increase in computer time.

Using the correct guess for a solution grid also increases the accuracy of a non-adaptive grid solution. Lange uses an optimized grid developed by Hodge, and the non-adaptive-grid solutions in this study use an optimized, power-law grid to solve the boundary-layer code (10). However, the adaptive grid solutions in this study use an exponential grid generated by a solution of the grid equation in the normal direction. Fig. 12 demonstrates the grid dependence of the boundary-layer code. The exponential grids are generated with Q equal to -.3. The 61x30 power law grid results are much better than the results for either a 61x30 or a 21x19 exponential grid. The type of grid chosen for the boundary-layer solution does affect the accuracy of the solution. The power law grid is accurate for boundary-layer problems since the boundary layer is close to a power-law shape except at the surface. But the exponential grid has the control function, Q, which can be varied depending on the characteristics of the flow solution. Optimizing Q yields an exponential grid which gives accurate results and does not require any foreknowlege of the exact solution.

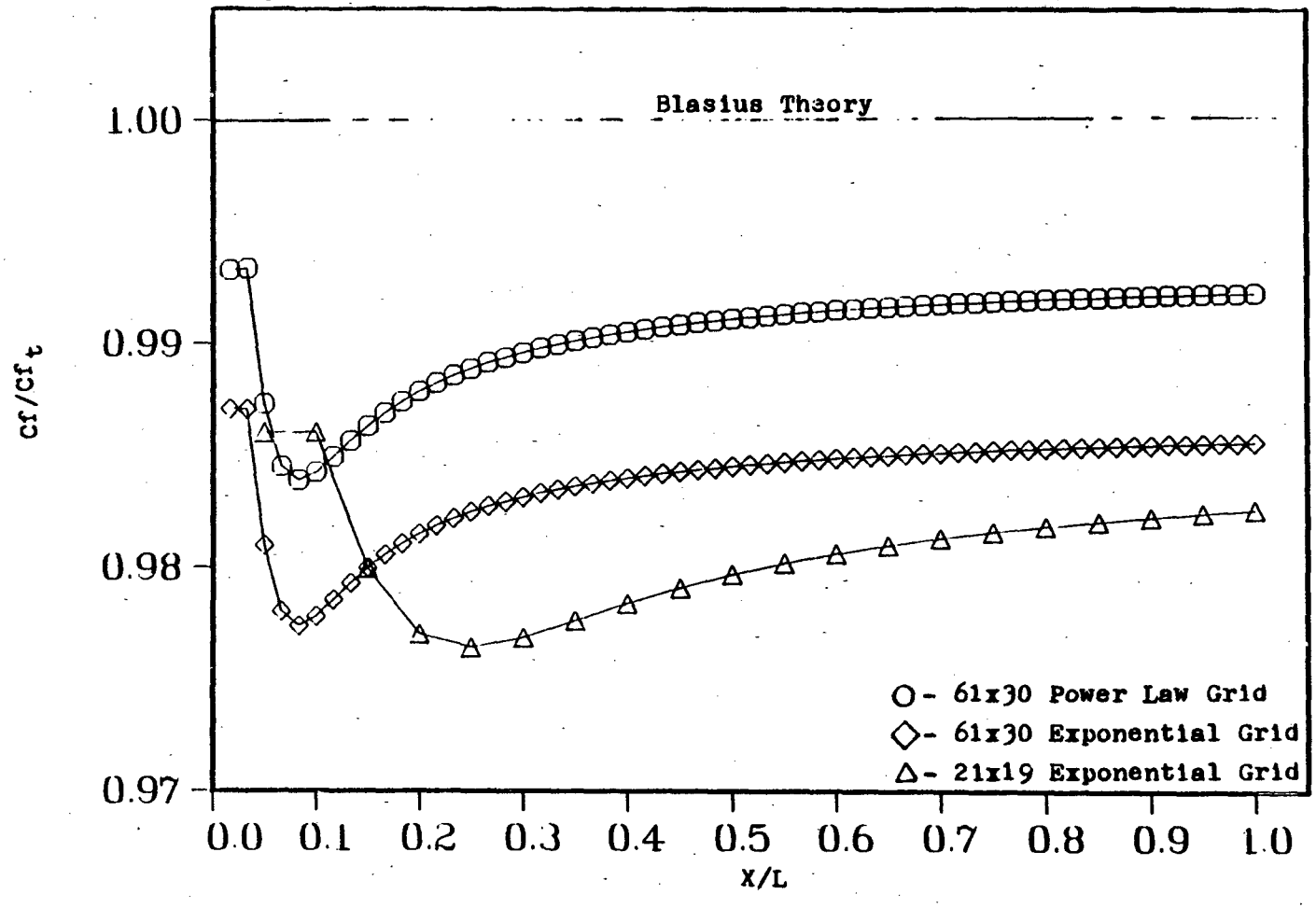


Fig. 12 Solution Comparison for Different Grid Types

Non-Adiabatic Wall Effects.

The test cases previously mentioned use input conditions which result in a nearly, adiabatic flow. The wall temperature is close to the adiabatic, wall temperature calculated from Eq (28). The approximately, adiabatic wall magnifies the effect of roundoff error for the calculation of h. The finite-difference solution uses Eq (55) to find h. The denominator in Eq (55) is very small for a nearly, adiabatic wall. Dividing q by a very, small number produces good results which give an h/h_{ref} near 1.0, but computer, round-off error causes the computed h to overshoot the values. Fig. 13 compares h/h_{ref} for the nearly, adiabatic wall case and a non-adiabatic case. The non-adiabatic case uses a wall temperature of 550°R, instead of 530°R as in the adiabatic-wall case. The two solutions use a 61x59 solution grid and an analytic definition of Y_ξ. The non-adiabatic case does not overshoot the theoretical solution, but it does have some error due to compressibility.

The non-adiabatic wall case highlights the behavior of the finite-difference solution toward h. With the non-adiabatic variation in wall temperature, compressibility effects are much greater than previous cases. The density ranges from .9963075 to .99941725 across the boundary layer. Therefore, there is some error in the solution due to this compressibility. Fig. 14 shows Cf/Cf_t is not affected much by the wall temperature change. However, h is affected. The non-adiabatic cases also make the solution for the total

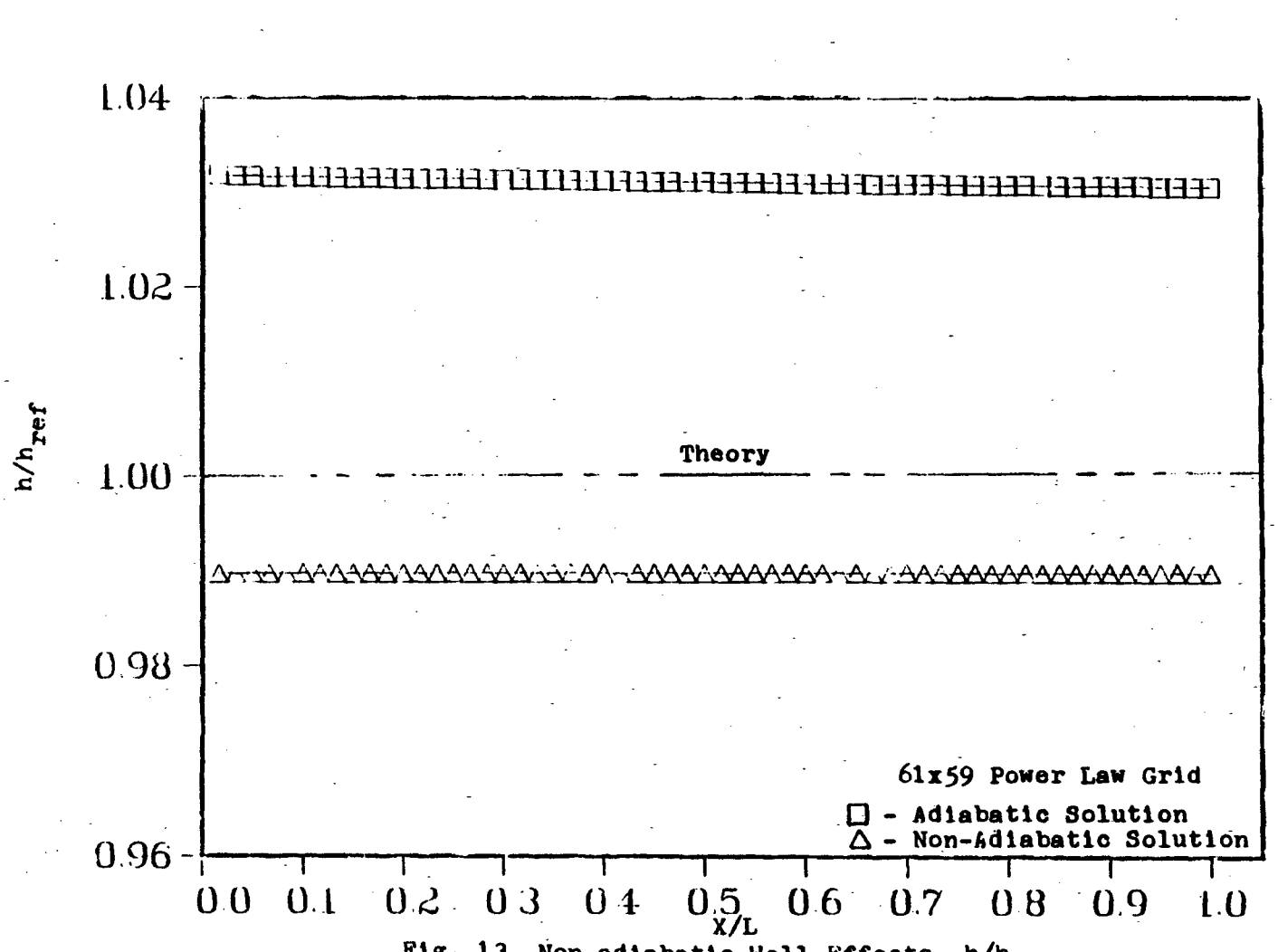


Fig. 13 Non-adiabatic Wall Effects, h/h_{ref}

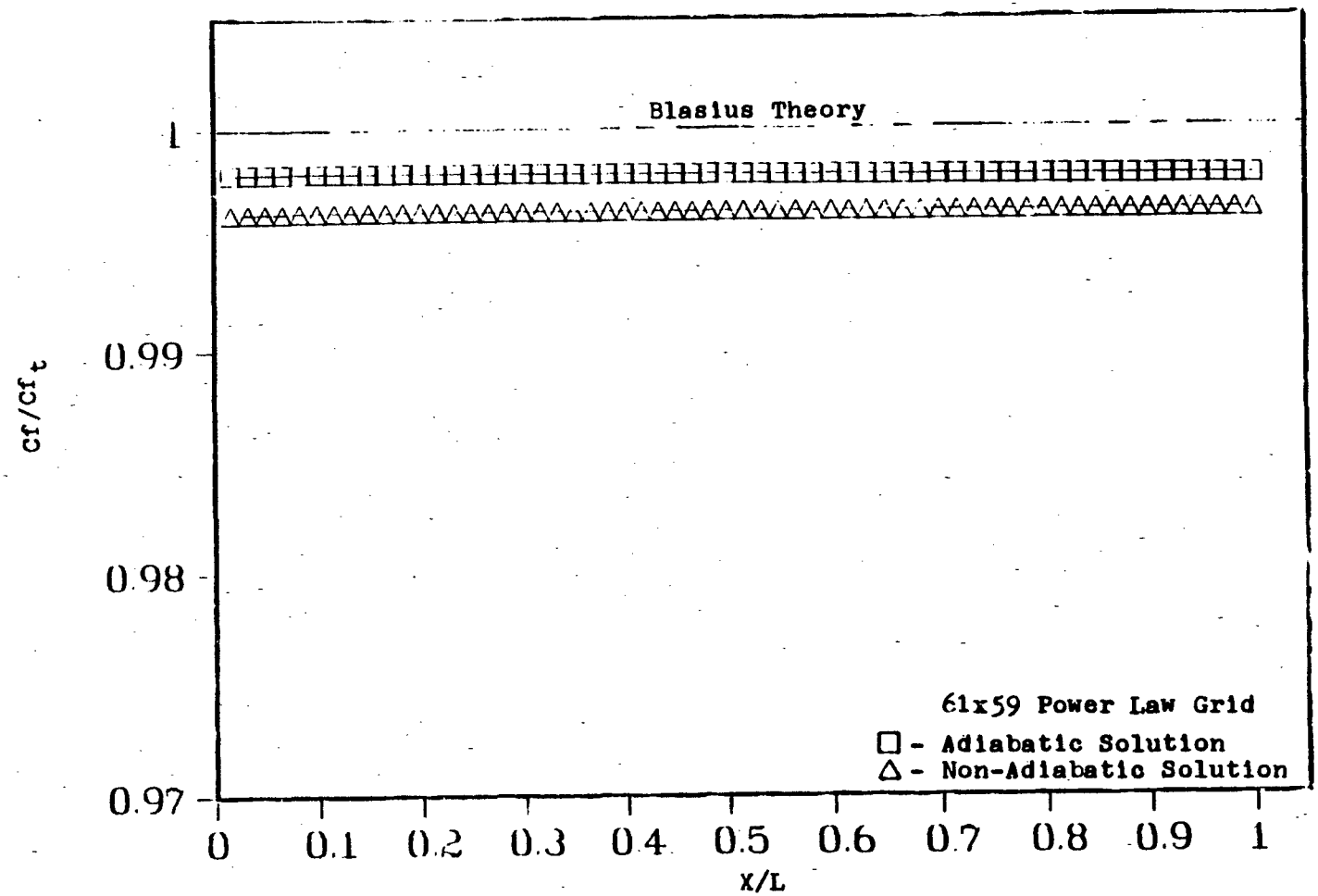


Fig. 14 Non-adiabatic Wall Effects, Cf/Cf_t

enthalpy much more difficult. For the adiabatic cases, it takes 20 iterations to converge to EPS of $1x10^{-6}$, and U was the usually the property that determined DELK. For the non-adiabatic cases, it takes over 80 time iterations to converge to EPS of $1x10^{-4}$. The finite-difference solution definitely has more trouble resolving H for the non-adiabatic cases. The finite-difference solution does converge to values close to the theoretical values, though. Using the same finite-difference solution with a Thomas-algorithm, iteration method, Hodge has computed ratios for Cf/Cf_t and h/h_{ref} of .9998 and 1.00, respectively (8). Therefore, the finite-difference solution of the incompressible, boundary-layer solution closely reproduces Blasius', exact solution.

Optimized Boundary Layer Code.

The adaptive-grid solution of the steady-state, boundary-layer problem finds the solution grid which best resolves boundary-layer flow. The finite-difference solution of the boundary-layer equations is very, grid dependent. If an optimized, solution grid is found, the accuracy of the computed solution increases. Using Powell's method to minimize the sum of the squares of the streamwise velocity gradients optimizes the grid and increases the accuracy of the computed solution. The increase in accuracy is more easily seen with a course grid, so the computer solutions use a 21x19 exponential grid. After showing that Powell's optimization method produces an optimized grid, this study tests

the use of different minimizing functions in the optimization. These functions are the sum of $U_\eta{}^2$, $U_{\eta\eta}{}^2$, or $U_{\eta\eta\eta}{}^2$. Minimizing some functions produces better optimization than others. The number of iterations, KT, and other input parameters such as the initial guess for Q also affect the optimization. If the initial guess for Q is too far off, Powell's method may not reach a converged solution in a small number of iterations. Without a converged solution, the optimized solution for the particular function may not be reached. Powell's method does minimize most of the input functions in very few, iteration steps.

Optimization Performance.

The computer solution generates the solution grid using the grid equation in Eq (35). The optimization finds the control function, Q, which generates a grid that minimizes the desired function and increases the accuracy of the boundary-layer solution. The flexibility of Powell's method allows the sum of $U_\eta{}^2$, $U_{\eta\eta}{}^2$, or $U_{\eta\eta\eta}{}^2$ to be minimized. Comparison of these three cases determines which minimization gives the best results. As in the non-adaptive case, a better solution comes closer to a value of 1.0 for Cf/Cf_t. In addition, the optimization which reduces the error in the computed solution is a better method. The root-mean-square (RMS) of the difference between the computed solution and the von Karman-Pohlhausen approximate solution measures the error in the computed solution. The different optimization solutions are also compared to the Blasius', exact solution.

With these comparisons, it is possible to discover how the adaptive grid is best used to get more, accurate, computed solutions to general, boundary-layer-type problems.

Table III summarizes test cases of the adaptive-grid solution for the incompressible flow over a flat plate. The input parameters for solution of the boundary-layer code within the optimization do not change. The only change in the boundary-layer code inputs is the solution grid calculated in the optimization. The boundary-layer code, input parameters are 20 time steps, 4 iterations per time step, a time step of $1x10^{6}$ seconds, error tolerance of $1x10^{-6}$, a Pr of .72, and recalculation of the SOR parameter every fifth time step. These are the same inputs as run 5 of the non-adaptive, incompressible solution. Optimizing 21x19 and 61x30 exponential grids is tested. Setting the minimum and maximum values of Y at the x location where the grid equation is solved determines the physical size of grid. These Y values are YMIN and YMAX in Table IIIA. The grid is then scaled in the streamwise direction with the Y spacing solved by the grid equation at the x location chosen. For this study, the grid equation is solved at the trailing edge of the surface. The parameter, Q, shows the initial input for the control function. The results of using a non-adaptive, exponential grid with the input Q's are given in Table IV. KT represents the number of iterations allowed to find each λ_r value. PCT is the percentage of λ_r added and subtracted to λ_r to get λ^+ and λ^-. The error tolerance between

TABLE IIIA

Adaptive Exponential Grid Solutions-Inputs*

Run	KT	PCT	YMIN	YMAX	Q	WH1	WH2	WH3
RE=2×10^6, Grid-21x19								
1	3	.2	0.0	0.015	-.3	0	0	1
2	6	.2	0.0	0.015	-.3	0	0	1
3	3	.5	0.0	0.015	-.3	0	0	1
4	3	.2	0.0	.0004	-.3	0	0	1
5	3	.2	0.0	0.030	-.3	0	0	1
6	3	.2	0.0	0.015	-.3	1	0	0
7	6	.2	0.0	0.015	-.3	1	0	0
8	6	.2	0.0	0.015	-.2	1	0	0
9	3	.2	0.0	0.015	-.3	.1	0	.9
10	3	.2	0.0	0.015	-.3	.5	0	.5
11	3	.2	0.0	0.015	-.3	0	1	0
12	3	.2	0.0	0.015	-.3	.25	.25	.5
13	3	.2	0.0	0.015	-.3	0	.5	.5
14	3	.2	0.0	0.015	.3	0	0	1
15	3	.5	0.0	0.015	.3	0	0	1
16	3	.2	0.0	0.015	-.6	0	0	1
17	3	.2	0.0	0.015	.05	0	0	1
RE=5×10^5, Grid- 21x19								
18	3	.2	0.0	0.030	-.3	0	0	1
19	3	.2	0.0	0.030	-.3	0	1	0
20	3	.2	0.0	0.030	-.3	1	0	0
21	6	.2	0.0	0.030	-.3	0	0	1
RE=5×10^3, Grid- 21x19								
22	3	.2	0.0	0.300	-.3	0	0	1

TABLE IIIA (cont.)

RE=2x10^6, Grid-21x10								
23	3	.2	0.0	0.015	-.3	0	0	1
24	3	.2	0.0	0.015	-.3	0	1	0
25	3	.2	0.0	0.015	-.3	1	0	0
RE=2x10^6, Grid-61x30								
26	3	.2	0.0	0.015	-.2	0	0	1

* : For boundary layer solution inputs see Run 3, Table II.

TABLE IIIB

Adaptive Exponential Grid Solutions- Results

Run	Cf/Cf$_t$ (leading edge)	Cf/Cf$_t$ (trailing edge)	F (initial-final)	RMS(U-Uexact) (initial-final)
1	.99135	.98781	.093644-.028557	.029654-.028736
2	.99135	.98781	.093644-.02255	.029654-.02836
3	.99135	.98781	.093644-.028557	.029654-.028736
4	.99589	.99590	.220564-.000611	.032002-.033511
5	.98473	.98394	72.308 - 70.934	.019075-.019598
6	.98624	.98281	2.2295 - 2.2073	.029654-.029671
7	.98636	.98292	2.22044-2.2107	.029654-.029671
8	.99039	.98687	2.0612 -2.0403	.029797-.029946
9	.99107	.98760	.30723 -.23333	.029654-.029764
10	.99053	.98707	1.16154-1.0384	.029654-.029925
11	.99139	.98735	.24103 -.10459	.029654-.028933
12	.99084	.98738	.66444 -.56026	.029654-.029861
13	.99138	.98784	.16734 -.063586	.029654-.028809
14	.64983	.64920	72.308 -71.503	.019075-.019381
15	.65324	.65259	72.308 -70.934	.019075-.019598
16	.94844	.94325	6.176 - 6.1275	.02993 -.029903

TABLE IIIB (cont.)

Run	Cf/Cf_t (leading edge)	Cf/Cf_t (trailing edge)	F (initial-final)	RMS(U-Uexact) (initial-final)
17	.99137	.98783	2.367 -.022564	.027888-.02877
18	.99135	.98785	.09365-.022259	.029654-.028732
19	.99135	.98785	.24102-.10455	.029654-.028834
20	.98686	.98292	2.229 - 2.203	.029654-.026815
21	.99135	.98781	.093654-.022544	.029654-.028736
22	.99135	.98781	.093655-.022544	.029654-.028736
23	.96920	.96591	.866760-.605740	.036949-.037135
24	.97324	.96990	.690502-.690086	.036949-.036962
25	.95975	.95656	4.38803-3.73679	.036949-.038336
26	.99237	.99378	.008120-.007551	.027758-.026635

TABLE IV

Non-Adaptive Grid Solutions*

Run	YMIN	YMAX	Q	Cf/Cf_t (lead.edge)	Cf/Cf_t (trail. edge)
Grid - 21x19 Exponential					
1	0.0	0.015	-.3	.89979	.90335
2	0.0	0.015	-.2	.91203	.91488
3	0.0	0.015	+.05	1.01017	1.04270
4	0.0	0.030	-.6	.89949	.89888
5	0.0	0.015	.3	.69791	.65252
Grid-21x19 Power Law					
6	0.0	0.0147	--	.98369	.98025

* : For boundary layer solution inputs see Run 5, Table II.

iterations of each λ_r value and between iterations of the overall optimization scheme are not shown in Table III.

These are set at $1x10^{-5}$. The last input parameters shown in Table III are the weighting parameters, WH1, WH2, and WH3. These weightings determine which velocity derivative, or combination of velocity derivatives, is the minimized function. The minimized function is represented by F and is defined by

$$F = WH1*\Sigma U_{\eta}^{2}+WH2*\Sigma U_{\eta\eta}^{2}+WH3*\Sigma U_{\eta\eta\eta}^{2} \qquad (66)$$

WH1 is the weighting on minimizing ΣU_{η}^{2}. WH2 is the weighting on minimizing $\Sigma U_{\eta\eta}^{2}$, and WH3 is the weighting on minimizing $\Sigma U_{\eta\eta\eta}^{2}$. The values in Table IIIB are the range of results showing the performance of the particular, adaptive-grid solution.

Powell's method minimizes the input function. As a result, the computed, boundary layer solution gets closer to theoretical values. This study seeks to minimize the computational error by minimizing truncation error. From an analysis of the truncation error terms, minimizing $U_{\eta\eta\eta}$ should reduce solution error. Therefore, this study concentrates on minimizing the input function, $\Sigma U_{\eta\eta\eta}^{2}$, with Powell's minimization method. Solution runs 1-4, shown in Table III, summarize the results of optimizing the grid with $\Sigma U_{\eta\eta\eta}^{2}$. The differences between the runs are the various input parameters. All of the solutions show a decrease in $\Sigma U_{\eta\eta\eta}^{2}$, represented by F, from the start of the program to its converged solution. Each of the solutions also comes closer to the theoretical solution than the initial grid.

The range of values for Cf/Cf_t is closer to 1.0 than similar results with either the initial, exponential grid or the non-adaptive, power-law grid solution shown in Table IV. Additionally, the optimized solution reduces the overall error in U. In run 1, the RMS of $U-U_{exact}$ across the whole domain is .029654 at the start of the minimization. The final RMS $U-U_{exact}$ value is .028736. This is a 3.096% decrease in the U error. Fig. 15 shows how close the solution comes to the exact u velocity profile over a flat plate. Therefore, the Powell's minimization method applied to the input function $\Sigma U_{\eta\eta\eta}{}^2$ reduces the error in U and increases the accuracy of the computed solution by minimizing the input function, F.

Although minimizing $\Sigma U_{\eta\eta\eta}{}^2$ meets the objective of reducing the error of the computed solution and increasing its accuracy, it may not be the only function which can meet these objectives. Using $\Sigma U_{\eta}{}^2$, $\Sigma U_{\eta\eta}{}^2$, or some combination of the three velocity gradients as minimizing functions produces different results than those optimizations on $\Sigma U_{\eta\eta\eta}{}^2$. Each of the different cases performs similar to minimizing $\Sigma U_{\eta\eta\eta}{}^2$ by decreasing F and increasing the solution accuracy. However, the error in U is not decreased in all cases. Minimizing $\Sigma U_{\eta}{}^2$ does nct decrease the error in U. Optimizing this function increases the error in U by .06%. Cf/Cf_t for this solution, run 6, is also below that of the runs 1-4. Since the function did decrease, an increase in the error is not expected. Closer examination of the solution shows the

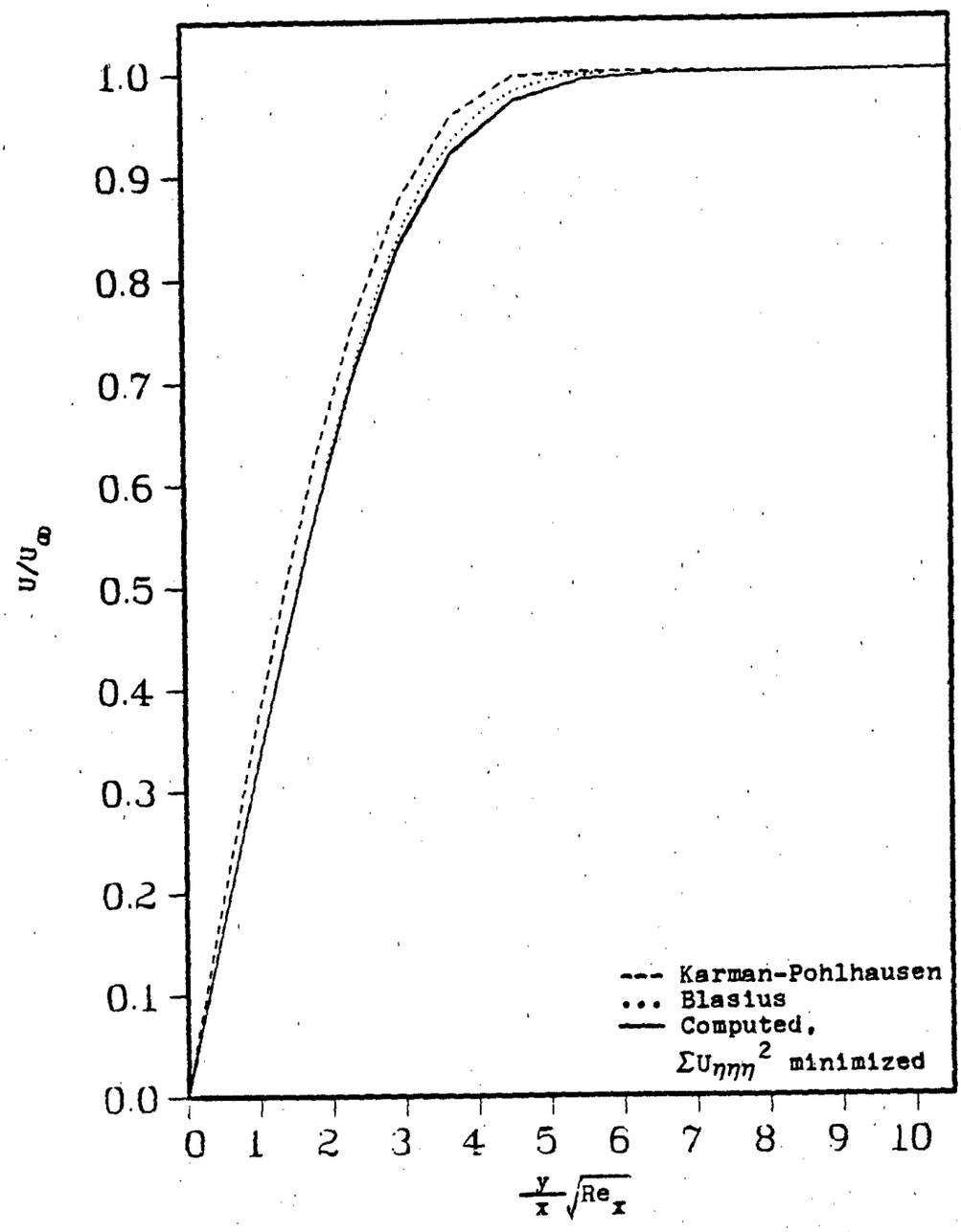


Fig. 15 Minimization of $\Sigma U_{\eta\eta\eta}{}^2$ - Velocity Profile

optimization is not totally converged. The solution was therefore rerun with a higher number of iterations. A Q closer to the results of previous runs was also input to help the solution converge faster. This solution, run 8 in Table III, did give better results for Cf/Cf_t. The results are still not as good as those in run 1. The error in U also increased by .496%. Minimizing $\Sigma U_\eta{}^2$ does keep more points in the boundary layer than other solutions. Run 1 keeps 11 points in the boundary layer. Runs 6-8 keep 14 points in the boundary layer. With more points in the boundary layer, the solution should resolve flow conditions better and produce better results. However, this does not occur. Fig. 16 shows the velocity profile produced by minimizing $\Sigma U_\eta{}^2$. It is close to the theoretical answer. However, Fig. 17 shows, the velocity profile solved by minimizing $\Sigma U_\eta{}^2$ is farther from the exact solution than either minimizing $\Sigma U_{\eta\eta\eta}{}^2$ or $\Sigma U_{\eta\eta}{}^2$. Regardless of the number of points in the boundary layer, the solution from optimizing $\Sigma U_\eta{}^2$ is not better than optimizing $\Sigma U_{\eta\eta\eta}{}^2$.

Optimizing with $\Sigma U_{\eta\eta}{}^2$ is better than optimizing $\sum U_{\eta\eta\eta}{}^2$ in some ways. Minimizing $\Sigma U_{\eta\eta}{}^2$ in run 11 gets almost as close to the theoretical results as runs 1-4. Cf/Cf_t starts out higher than case 1 but recovers to values lower than run 1. The decrease in error for U is only 2.766%. Minimizing $\sum U_{\eta\eta}{}^2$ resolves the leading edge gradients better to get better values at the leading edge. But it does not produce better results at the trailing edge where the leading edge

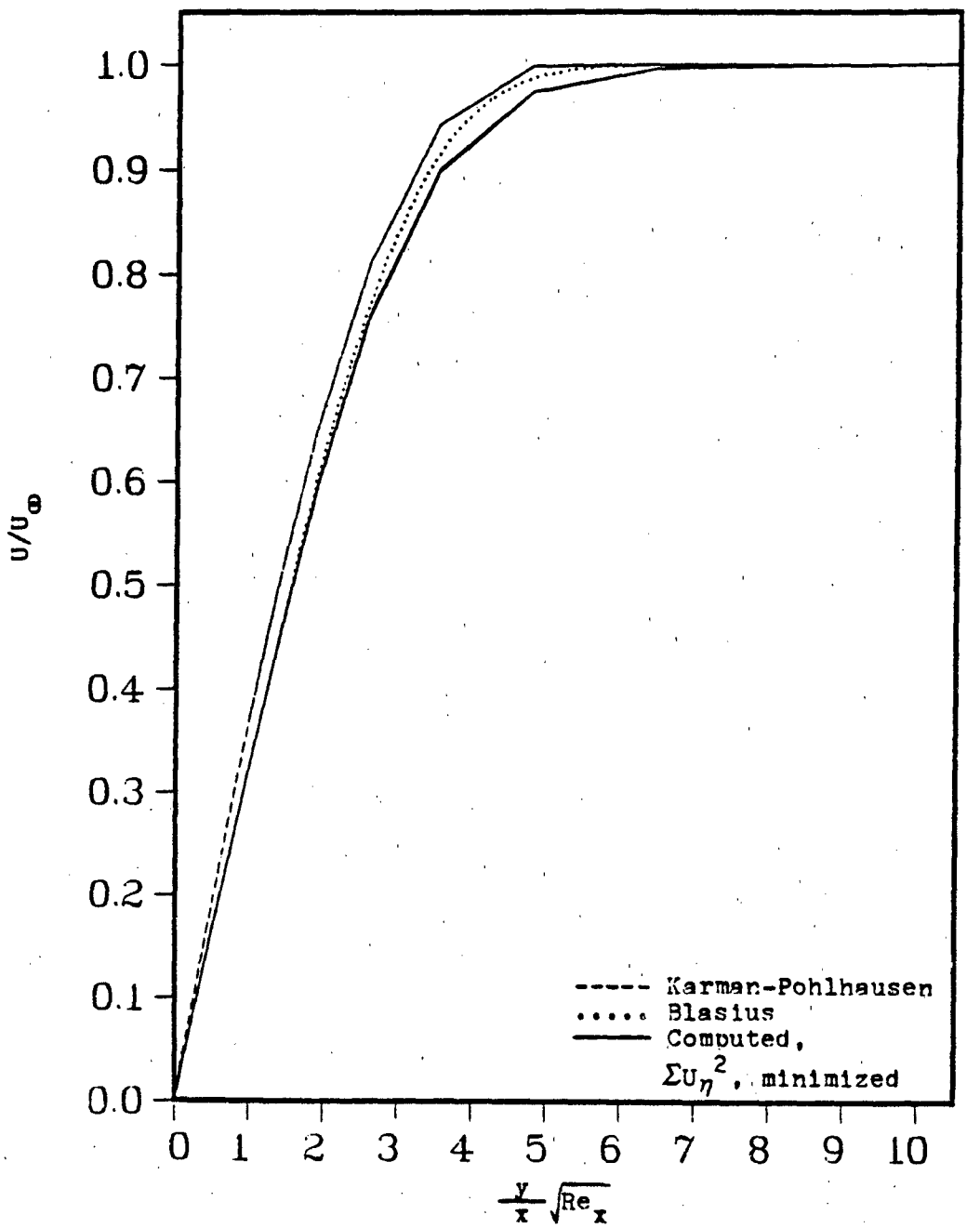


Fig. 16 Minimization of $\Sigma U_\eta{}^2$ - Velocity Profile Comparison

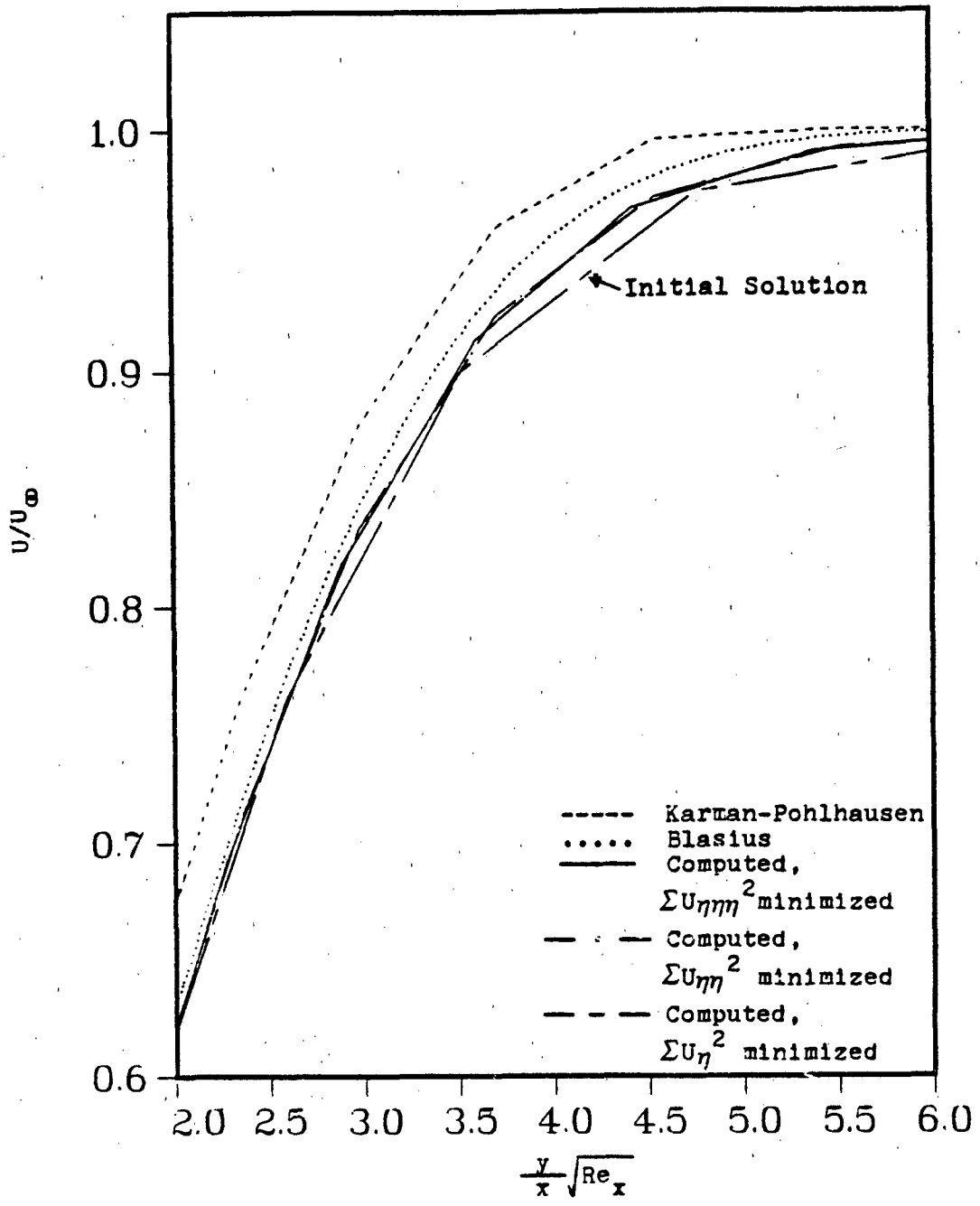


Fig. 17 Minimized Function Effect on the Velocity Profile

error does not affect the solution as much. Over the entire domain, the sum of $U_{\eta\eta}{}^2$ does not reduce the error in U as well as minimizing the sum of $U_{\eta\eta\eta}{}^2$. Fig. 17 shows that minimizing $\sum U_{\eta\eta\eta}{}^2$ minimizes the error in U better and produces a velocity profile closer to the exact solution than minimizing either the sum of $U_{\eta\eta}{}^2$ or the sum of $U_{\eta}{}^2$.

Using each of the three, different, velocity gradients separately as the minimizing function in Powell's method shows the sum of $U_{\eta\eta\eta}{}^2$ to be the best minimizing function. Using combinations of the three velocity gradients also does not improve the solution. In runs 6-13, different weightings are applied to the different gradients. The weighting determines how much a part of the minimized function the specific gradient is. When $\sum U_{\eta}{}^2$ is any part of F, the velocity error increases and the computed solutions are worse than run 1 where only $\sum U_{\eta\eta\eta}{}^2$ is minimized. Combining the sum of $U_{\eta\eta}{}^2$ and $U_{\eta\eta\eta}{}^2$, as in run 13, gives slightly better results for Cf/Cf_t. In the combination, the $\sum U_{\eta\eta}{}^2$ contribution resolves the leading edge error while $\sum U_{\eta\eta\eta}{}^2$ raises the trailing edge values. The combination of the two functions thus produces a grid which gets the closest values to the Blasius', theoretical solution. The error in U only decreases by 2.85%. Minimizing $\sum U_{\eta\eta\eta}{}^2$ still reduces the error in U better than any of the functions tested. There may be some combination of the $\sum U_{\eta\eta}{}^2$ and $\sum U_{\eta\eta\eta}{}^2$ not tested which may reduce the error in U better in addition to computing better results for C_f.

Several of the input parameters affect the behavior of

the optimization method. Inputs KT, PCT, and Q affect the convergence of the method, and YMAX affects the accuracy of the solution. Increasing KT only affects those optimizations which need many iterations to converge. Powell's method has many levels at which it iterates. The iteration of λ_r and iteration of the total solution are two of the levels. Although a problem may fail to converge at one level, succeeding levels take out this error to converge with a small KT. Using a large KT for quickly converging problems converges the solution in the iteration of λ_r. All steps which follow merely check this convergence. This checking uses computer time but adds nothing to the solution. The latter steps of the method are therefore wasted. This occurs in run 2. However, for slow converging problems, increasing KT is essential to reach converged answers. Optimizing on $\sum U_\eta^2$ requires many iterations. For example, run 6 does not converge with a low KT. Runs 7 and 8 converge with only double the KT of run 6. The results of the converged problem are better than those of the unconverged problem. In run 8, the initial Q is changed to help convergence. If the initial Q is very far away from the final, optimum Q, the solution needs more iterations to converge. Runs 14 and 15 are solutions with an initial Q of .3. From solutions 1-3, the optimum Q for the same minimized function and grid size is -.144. In runs 14 and 15, Q only decreases from .3 to .297 in three iterations. This solution is not converged. The solution grid resulting is not optimized, as shown by the

values of Cf/Cf_t. When the initial Q is changed to .05 in run 17, the solution converges to -.144 the same as runs 1-3. This shows positive guesses for Q can converge, but the initial guess must be relatively close to the optimum Q for a small KT. Large, negative guesses for Q also do not converge. With Q initially set at -.6, the solution converges on Q of -.594. This is a long way from the run 1 solution of -.144. Negative, initial guesses for Q cannot be too far away from the final Q. There is a another limit on negative, Q values. Q cannot at any time in the iterations be allowed to get too small for proper resolution of U in the finite difference. In the 21x19 grid, this Q is -.8. For a 61x59 grid, Q of -.3 compresses the grid too much. The U velocities for a very, compressed grid are virtually the same for the y locations next to the wall. Therefore, the velocity gradients go to zero, and the solution diverges. Large, negative guesses for Q should be chosen carefully. Choosing a large PCT also can cause the solution to diverge. PCT determines how far away from λ, λ^+ and λ^- are. This optimization method performs like any iterative scheme. If guessed values are too widely spaced, more iterations are required to converge to a solution , or the solution may even diverge. Run 3 which uses PCT of .5 converges on the same answer as run 1. Run 1 converges in 6 iterations, but run 3 converges in 9. The larger PCT is therefore not as efficient, unless the initial guess is very far away from the optimum Q. Choosing PCT, KT, and the initial Q correctly leads to a

converged answer, but the size of the physical grid affects the accuracy of the solution. Choosing YMIN and YMAX determines the physical size of the grid. The exponential grid is built within these limits. For most of the solutions, YMAX is .015. This is approximately 4δ for Re of $2x10^6$. In run 4, YMAX is decreased to .004, or approximately δ. This puts the majority of the points inside the boundary layer. The optimization solves the boundary layer better when only the boundary layer is included in the domain. As expected, solution 4 is very close to the Blasius' solution. If YMAX is expanded to 8δ as in run 5 , the solution is not as good as solutions with thinner, solution grids. Therefore, the choice of YMAX does affect the optimized solution. If solutions are made with variable, flow conditions, YMAX should be varied to keep the same 4δ grid size in the y direction. Otherwise, the results are not compatible.

Various, input parameters and minimizing different functions affects the optimized solution. Changing the flow conditions or the number of grid points does not change the behavior of the solution for given minimizing functions. In runs 18-22, putting in different Reynold's numbers for cases minimizing $\sum U_{\eta\eta\eta}{}^2$, $\sum U_{\eta\eta}{}^2$, or $\sum U_{\eta}{}^2$ tests the effect of different, flow conditions. The solutions from runs 18, 19, 20, or 21 are no different at Re of $5x10^5$ than the corresponding runs 1, 6, 11, or 2 : Re of $2x10^6$. For minimizing $\sum U_{\eta\eta\eta}{}^2$, there is no difference for any of the solutions in runs 1, 2, 18, 21, or 22. Changing Reynold's number has no affect on

which minimizing function optimizes the solution best.

Changing the number of grid points alters the accuracy of the solution. In run 26, a 61x30 grid definitely gives better values of Cf/Cf_t than the 21x19 grid used in run 1. This is expected. Decreasing the number of grid points decreases the accuracy of the solution. Runs 23-25 are two to three percent less accurate than similar runs 1, 7, and 11 with the 21x19 grid. However, using the courser, 21x10 grid changes the relative accuracy of the three minimizations procedures. Minimizing the sum of $U_{\eta\eta}{}^2$ gives better results for the course grid than does minimizing the sum of $U_{\eta}{}^2$ or $U_{\eta\eta\eta}{}^2$. The optimum value of Q also changes with the number of grid points. A larger number of grid points requires an optimum Q closer to 0. The 21x10 grid in run 23 converged on a Q of -.33361. The 21x19 grid in run 1 converged on a Q of -.14426, while the 61x30 grid in run 26 required an optimum Q of -.0848. As the number of grid points increases, the optimum solution comes closer to a uniform grid. Therefore, the number of grid points does affect the accuracy and the method which best optimizes the boundary-layer solution.

Powell's method optimizes the incompressible, boundary-layer solution. The sum of $U_{\eta\eta\eta}{}^2$ is the best minimizing function to minimize the error in the streamwise velocity. Parameters, YMAX, ET, PCT, and the initial Q determine how fast and what values the solution converges on. Changing the Reynold's number of the flow problem does not affect the behavior of the optimization. Increasing the number of grid

points does increase the accuracy of the solution and does change the optimization behavior for course, solution grids. Incompressible, flow problems can be solved more accurately with an adaptive grid using fewer, grid points. The optimization can also be applied to resolve more variable, compressible, or turbulent, boundary-layer problems.

Compressible Flow

The behavior of the optimization and boundary-layer codes has already been examined for the incompressible, flow problems. Since the programs were initially set up to handle compressible, flow problems and had to be fooled to treat incompressible flow, the extension of the optimization to supersonic and hypersonic, compressible, flow problems is an easy step. The two, largest problems are defining an initial grid for the finite-difference solution and comparing the results with previous, theoretical data. In the incompressible cases, the boundary-layer thickness determined the physical size of the computational domain. YMAX, the height of the grid at the surface's trailing edge, was set at approximately 4δ to get consistent results. For the compressible cases, the boundary layer is not constant. The physical size of the boundary layer changes with flow velocity. Also, the transformed, boundary-layer thickness used with the compressibility transformation changes with the ratio T_w/T_e, not with the velocity. Therefore, as the compressible solutions are applied to different velocities and temperature condi-

tions, some method of defining a consistent, transformed-grid size must be found. The other problem with the compressible cases is determining the accuracy of the computed solution. There is no exact solution for compressible flow over a flat plate or wedge. There is no equivalent to the incompressible, Blasius' solutions. Eckert theory predicts approximate values for C_f and h across a high-speed flow. Also, Van Driest's study of compressible, boundary-layer flow over a flat plate provides a theoretical development to compare with the computed results. These comparisons show the validity and limitations of the boundary-layer code.

The major difference between the compressible and incompressible, boundary-layer solutions is the activation of the compressibility transformation. As flow velocity increases and temperature variations occur across the boundary layer, compressibility affects the flow more and cannot be ignored. This study assumes compressibility effects to be important beginning at about Mach .05. Above Mach .05, the Dorodnitsyn transformation applies to the problem. The problem is set up with the same equations as the incompressible problem. However, a new, boundary-layer thickness, $\Delta(X)$, is calculated with Eq (16). Above the transformed, boundary layer, the flow is inviscid. Initial conditions are the same as the incompressible case, except that the initial thermal boundary-layer thickness is assumed equal to the transformed boundary-layer thickness. Since $\Delta(X)$ varies depending on the edge conditions, differing Mach number, Reynold's num-

ber, or temperature conditions require a variable definition on the physical size of the solution grid. In the incompressible cases studied, YMAX, the height of the solution grid at the surface's trailing edge, is a fixed input. For the compressible cases, YMAX varies with the size of $\Delta(X)$. It can be set at any multiple of $\Delta(X)$ depending on the requirements of the flow problem. The effect of using different multiples of $\Delta(X)$ on the solution is shown in Fig. 18. This figure shows the computed, velocity profile at the trailing edge for different, solution methods. Similar to the incompressible cases, the closer YMAX is to $\Delta(X)$ the better the solution is to theoretical results. The solution for YMAX at $2\Delta(X)$ is closer to the theoretical result than $3\Delta(X)$ or $4\Delta(X)$ solutions. The application of the Dorodnitsyn transformation leads to computed solution which behaves like the incompressible solutions in other ways, also.

Although the compressible, boundary-layer problem does not have an exact solution similar to the Blasius' solution for flat plate flow, this study compares the computed solution to theoretical results obtained by E. R. Van Driest. Van Driest solves the laminar, compressible, boundary-layer equations for flow problem over a flat plate using Crocco's method (24:1). Van Driest's conditions on the flow problem are

> Prandtl number was taken at 0.75, the specific heat constant, and the Sutherland's law of viscosity-temperature variation was assumed to represent viscosity data starting with an ambient temperature of $-67.6^{\circ}F$ (24:1).

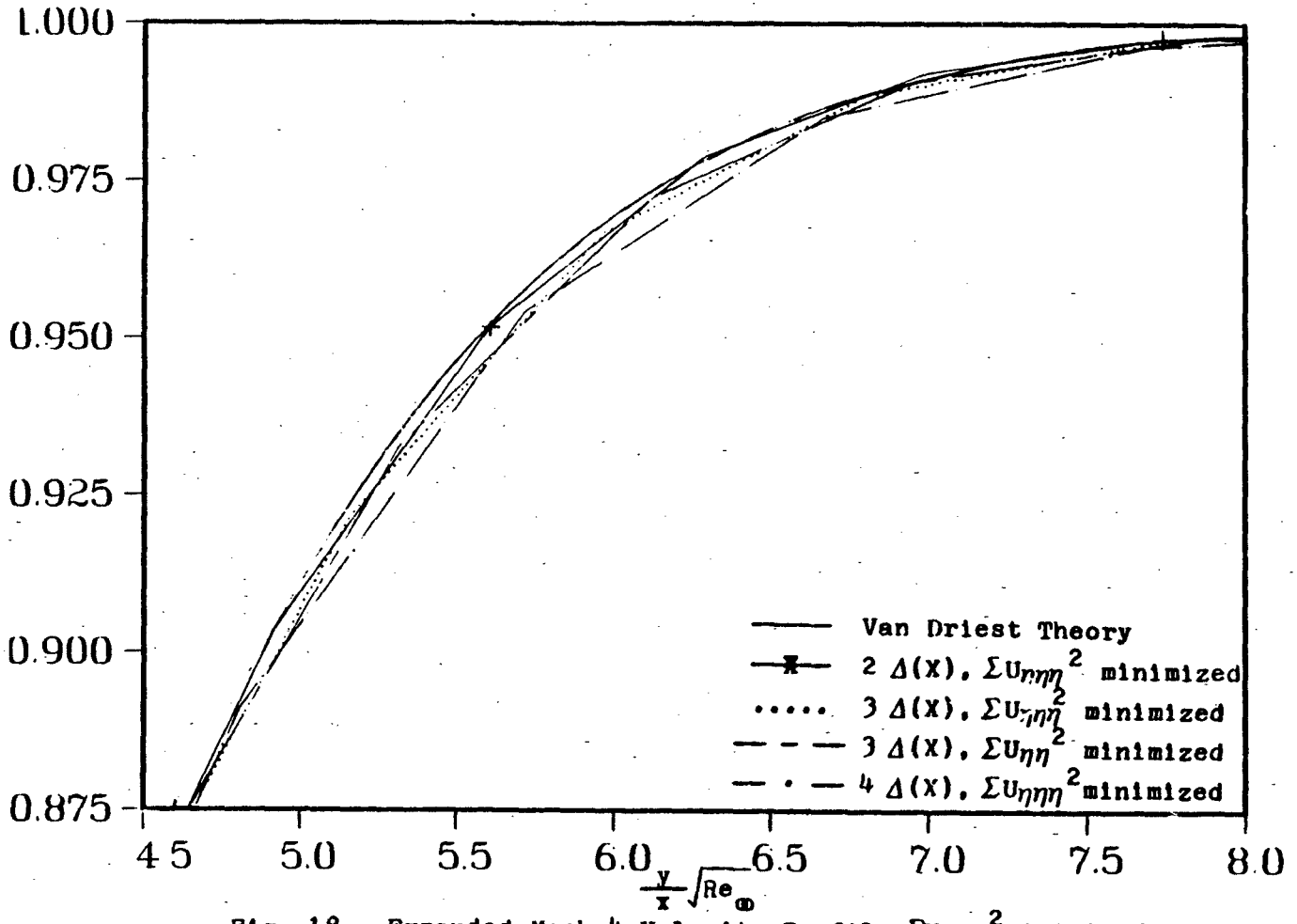


Fig. 18 Expanded Mach 4 Velocity Profile-$\Sigma U_{\eta\eta\eta}{}^2$ minimized

Van Driest's assumptions coincide with approximations made in this study. Van Driest solves the flow problem for many, different, flow parameters. Rather than show a comparison with the extensive results of Van Driest's development, this study examines a few cases to compare computed solutions to Van Driest's, theoretical solutions.

The computed solutions are very close to Van Driest's solutions. The cases solved are for Pr of .75, the ratio of Sutherland constant, 198.6°R, to free stream temperature at .505, and T_w/T_∞ at 1. This study tests supersonic and hypersonic cases for Mach 2 to 8. Figs. 19 and 20 compare the computed velocity and temperature profiles across the plate for the Mach range. These solutions set the upper limit of the domain at $3\Delta(X)$ away from the plate and optimize on the sum of $U_{\eta\eta\eta}{}^2$. The computed, velocity profiles are very close to Van Driest's profiles. In fact, the Mach 4 theoretical velocity profile lies exactly over the computed solution. Fig. 21 shows several, different, optimized solutions for Mach 4. All of the optimizations are so close that an expansion of the graph is needed to see any difference in the solutions. Fig. 18 shows the Mach 4 velocity profiles that result from optimizing on the sum of $U_{\eta\eta\eta}{}^2$ and on the sum of $U_{\eta\eta}{}^2$. Optimizing with the sum of $U_{\eta\eta\eta}{}^2$ is slightly better than optimizing with the sum of $U_{\eta\eta}{}^2$. This is consistent with the results of the incompressible cases. Although the Mach 8, velocity profile shows more error than the Mach 4 case, the Mach 8, velocity profile is also close. Since this

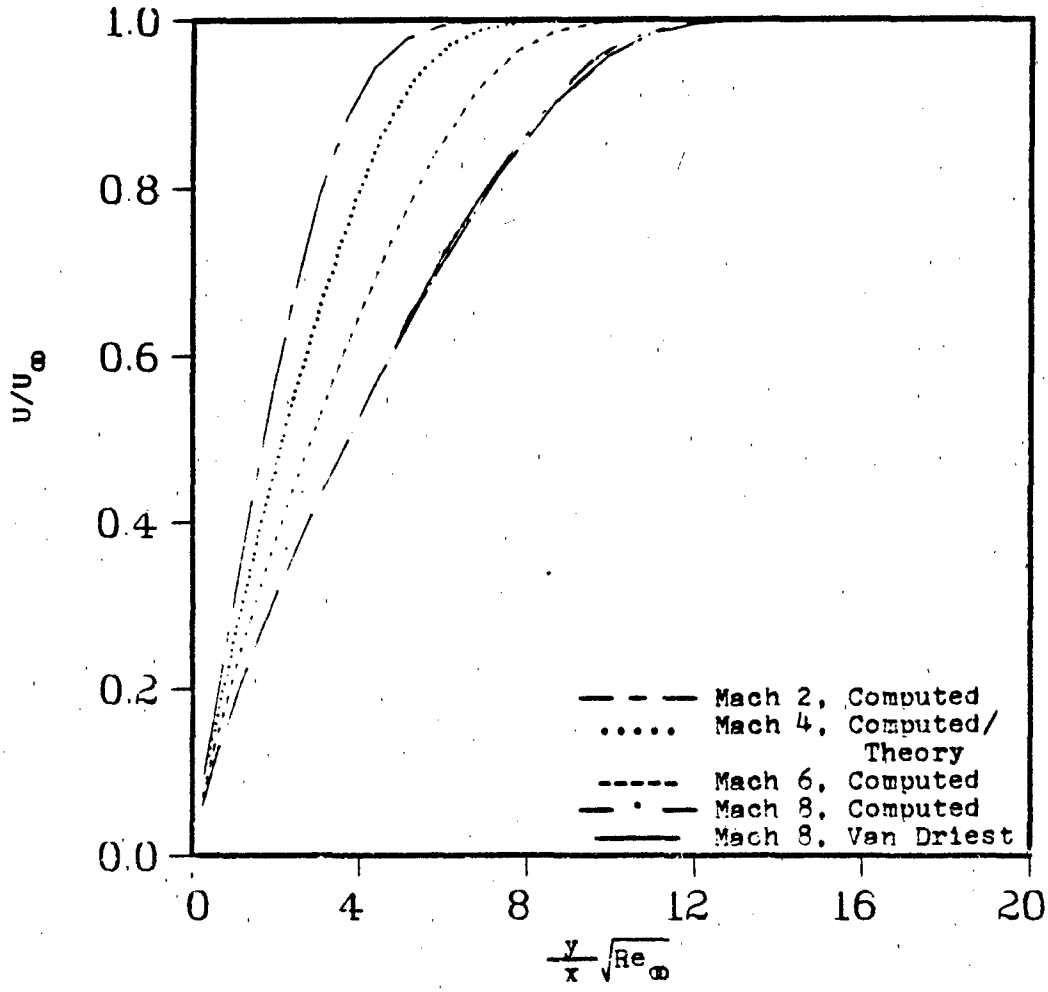


Fig. 19 Velocity Profiles- $T_w/T_\infty = 1.0$, $\Sigma U_{\eta\eta\eta}{}^2$ minimized

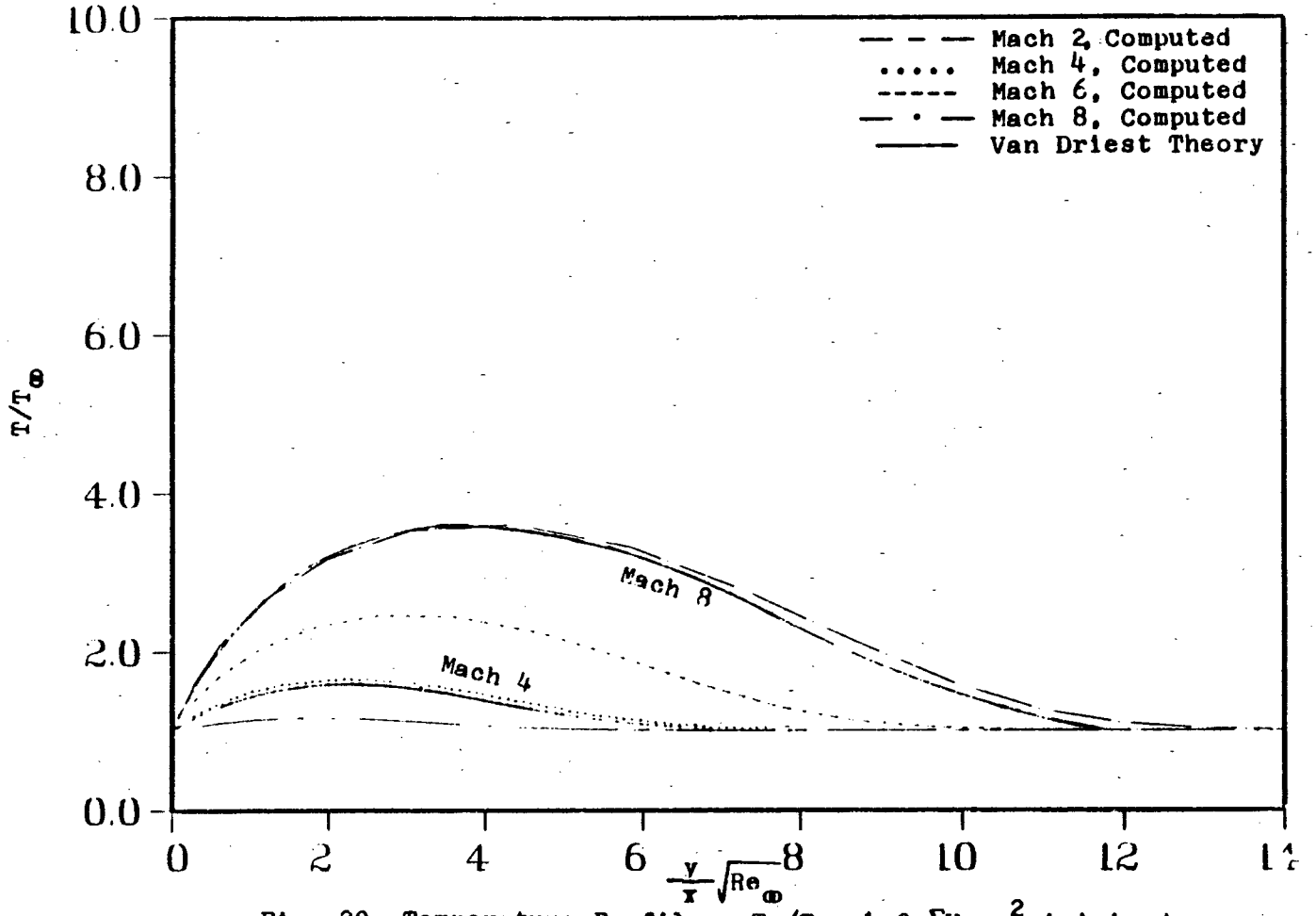


Fig. 20 Temperature Profiles- $T_w/T_\infty = 1.0, \Sigma U_{\eta\eta\eta}{}^2$ minimized

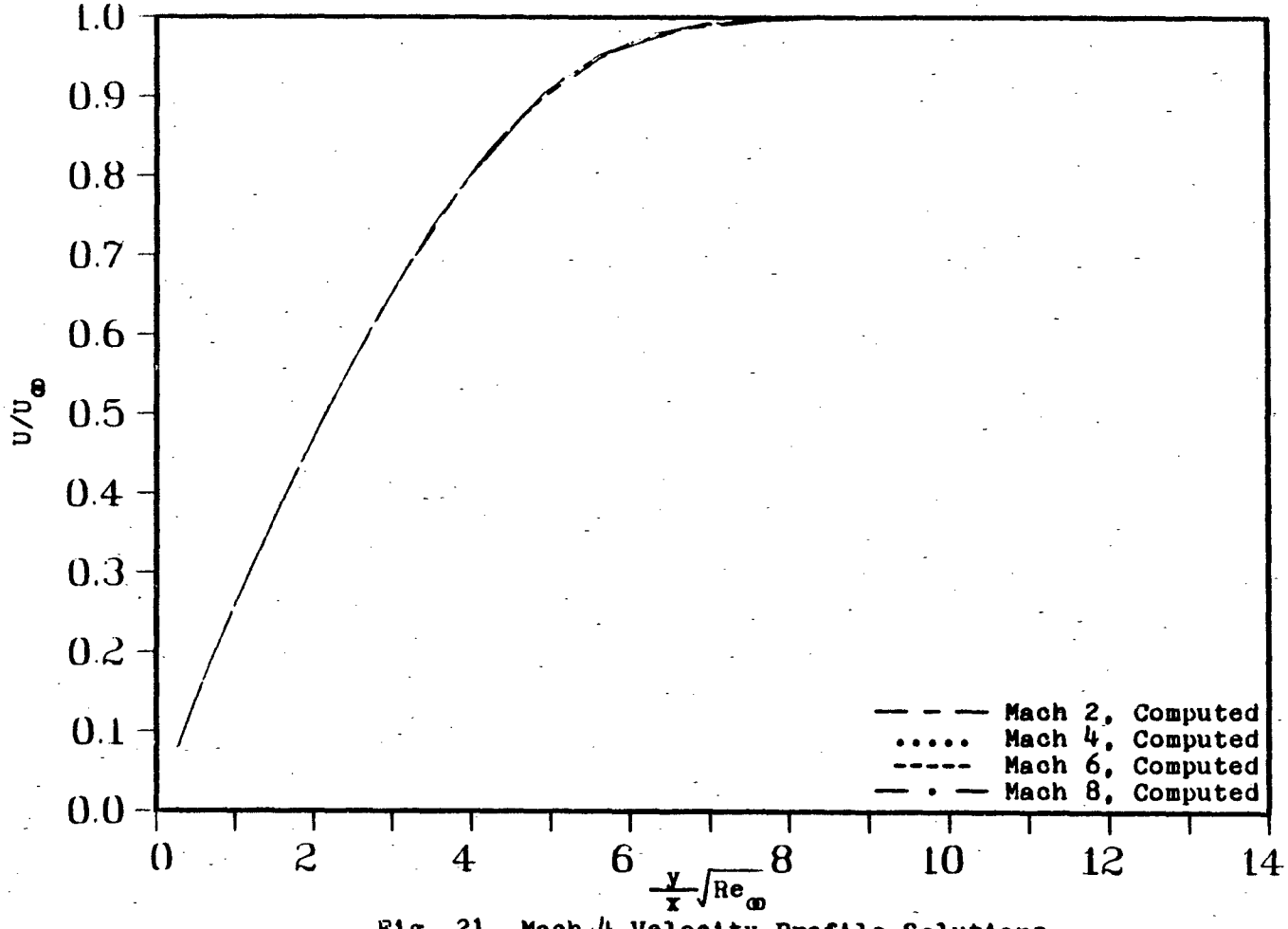


Fig. 21 Mach 4 Velocity Profile Solutions

study is restricted to flows where a perfect-gas assumption is valid, some variation from the Van Driest's, theoretical model is expected for Mach numbers above 6. The temperature profiles in Fig. 20 are not quite as exact. The computed, temperature solutions are less than the theoretical solution at some points. Since the boundary-layer code reduces the error in U by minimizing the sum of $U_{\eta\eta\eta}{}^2$, the computed solution should be close to the theoretical solution. However, the temperature profile results show the code does not necessarily produce optimized temperature results for Prandtl number not equal to one. It does achieve the goal of reducing the error in U.

The computed results also come close to values of h predicted by high-speed Eckert theory. For high-speed problems, Eckert recommends using an approximate temperature, T^*, to calculate an approximate density, ρ^*, and viscosity, μ^*, for the compressible, boundary layer. Eq (33) gives a value for T^*. The approximate, heat convection coefficient, h^*, is then calculated by putting the * conditions in Eq (32) as an approximation for h across the compressible, boundary layer. Eq (32) only applies to laminar flow with Re not greater than $5x10^5$ (12:213). By restricting flow conditions to Re less than this limit, the computed results for h come close to Eckert's, high-speed predictions. Fig. 22 shows the computed solutions for Mach 2, 4, 6, and 8 relative to Eckert's solution. A solution at Mach 14.24 over a 20^o wedge is also included to show the solution does work over a wedge.

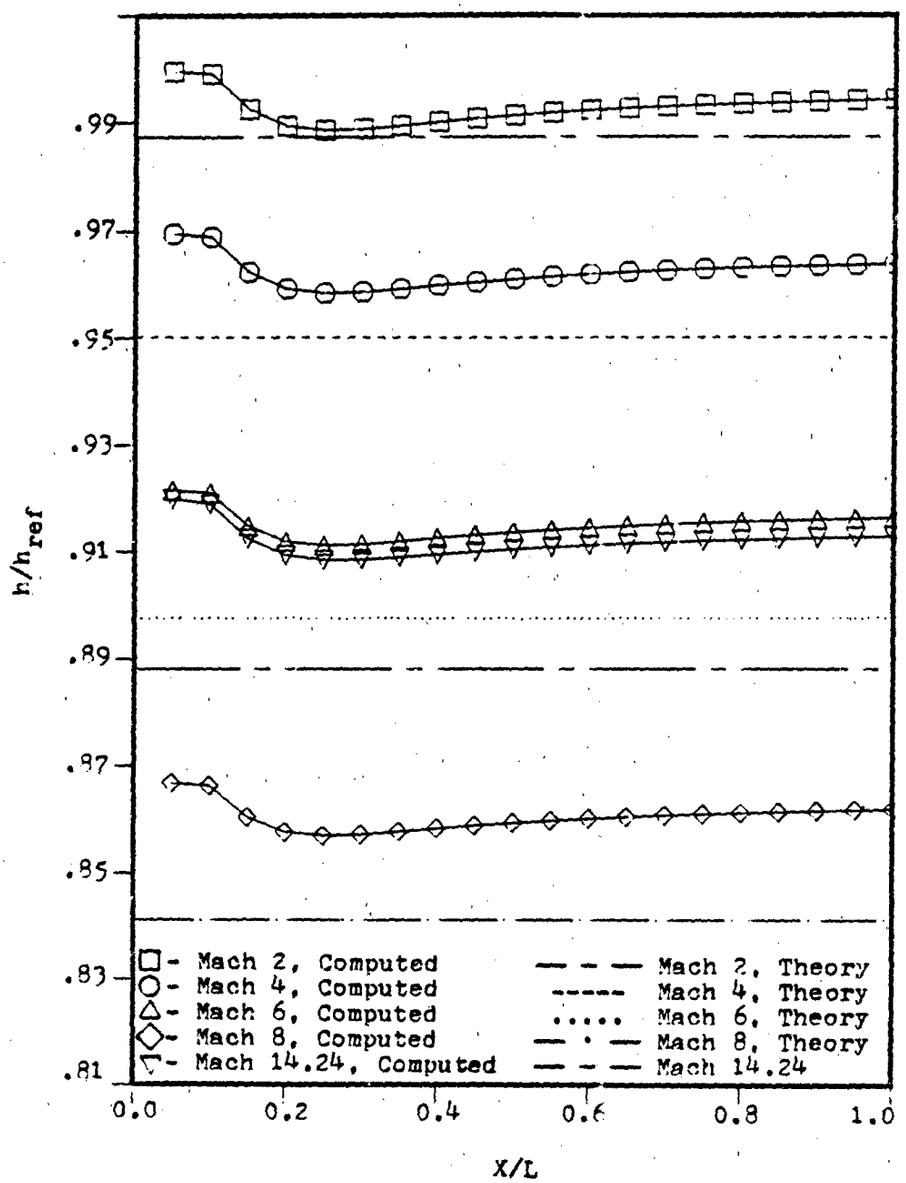


Fig. 22 Convective Heat Coefficient- $\Delta(X)$, $\Sigma U_{\eta\eta\eta}{}^{2}{}_{min}$

Both the theoretical and computed solution are divided by h_{ref}, the heat convection coefficient calculated for free-stream conditions. All of the solutions shown in Fig. 22 optimize the solution grid with the sum of $U_{\eta\eta\eta}{}^2$ over $3_\Delta(X)$. If a $1\Delta(X)$ solution grid is used, the computed h should get closer to Eckert's theory. Fig. 23 shows this is not the case. Fig. 23 presents h/h_{ref} results for Mach 4 with YMAX at different multiples of $\Delta(X)$. The increase in h/h_{ref} for $1\Delta(X)$ solutions results because less than the true, transformed, boundary-layer thickness is included in the domain. The initial guess for the boundary-layer thickness, $\Delta(X)$, is too small. The computed solution actually uses about $1.5\Delta(X)$ as the point where inviscid flow occurs and U_e/U_{inf} is one. Comparisons of the results in Fig. 23 indicate that the closer YMAX is to the true, boundary layer, the better the results for h are. However, the entire boundary layer must be included. Since the initial guess is not precise, it is difficult to pick the YMAX position to get only the boundary layer. Including too much of the inviscid flow above the boundary layer also dilutes the boundary-layer results. Some problems may require the solution of more than the flow inside the boundary layer. When the problem is set up, the desired accuracy and size of the problem domain must be considered for the best results possible. The optimized, boundary-layer solution provides the most flexibility for dealing with compressible, laminar, boundary-layer, flow problems and still achieving results close to theoretical predictions.

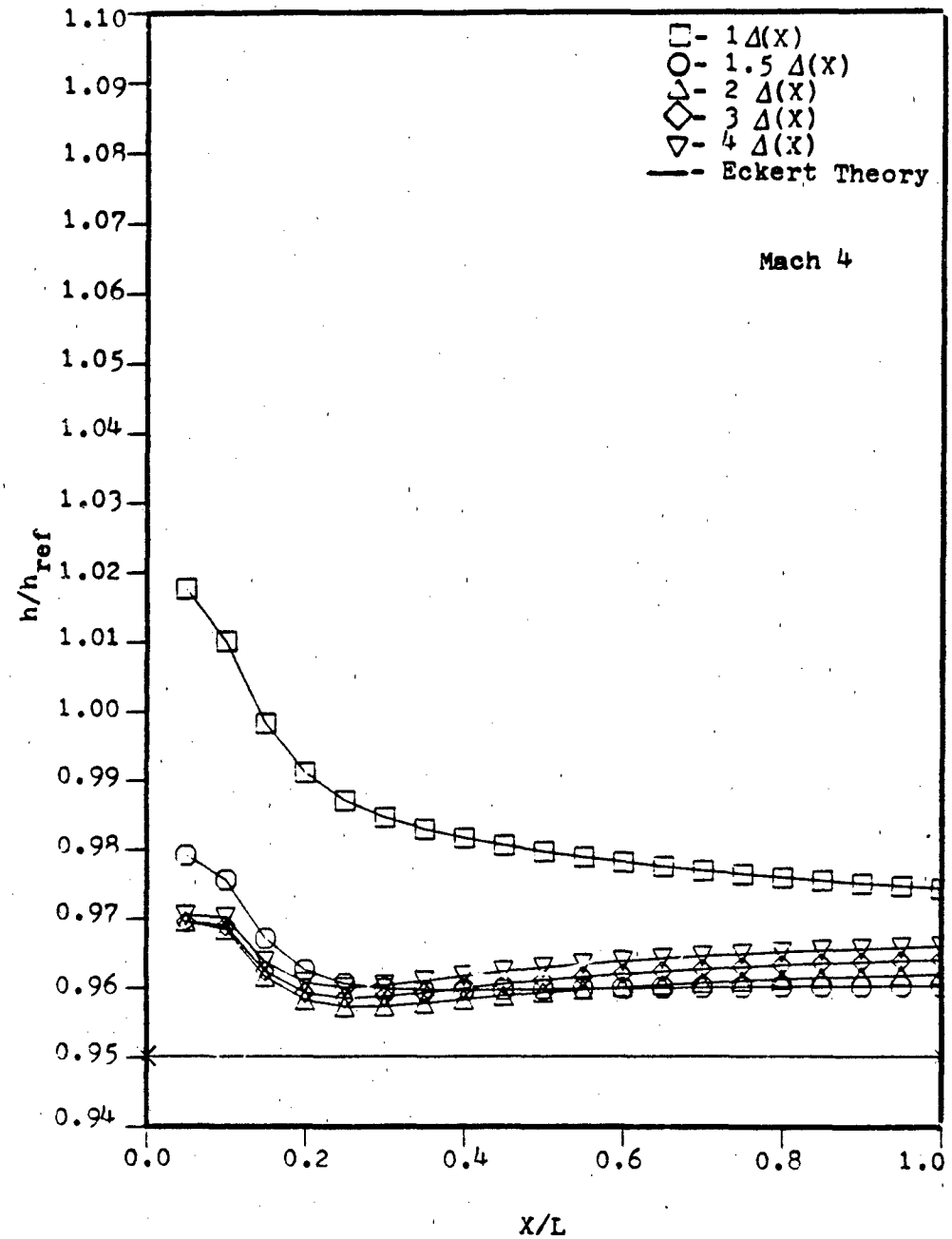


Fig. 23 Domain Thickness Comparison- $\Sigma U_{\eta\eta\eta}{}^2$ minimized

Chapter VI: Conclusions

In incompressible, flow problems, the theoretical solution gives a clue that the solution grid should be a power-law grid. For most flow problems, however, theoretical solutions do not exist to give a guess for the solution grid. Being able to input a general, exponential grid, which can be optimized to give the best flow solution, allows the most general application of a boundary-layer code.

The boundary-layer code presented in this study is an improvement over Lange's, boundary-layer code. For the non-adaptive solutions, the modifications made to Lange's code improve the solutions. Non-adaptive solutions for C_f calculated by the new code are closer to the Blasius', exact solution for flow over a flat plate. In addition, the new code includes a minimization method which optimizes an adaptive grid to find better solutions of boundary-layer problems.

Optimizing the adaptive grid with Powell's method produces more accurate, boundary-layer solutions, but the inputs must be chosen carefully to get the best optimization. The boundary-layer solution is very, grid dependent. The number of grid points, physical size, and type of grid definitely affects the results of the finite-difference solution. In both non-adaptive and adaptive cases, the solution with the largest number of grid points gives the best results. The problem whose domain includes less of the inviscid region

outside the boundary layer produces the better results. For example, problems that included only 1δ in the normal direction had better results than problems that used 4δ as the height of the grid. More grid points are in the boundary layer with only 1δ as the domain limit. In the non-adaptive cases, the power-law-type, solution grid gives better results than an arbitrary, exponential grid. However, the power-law grid only applies to a specific, boundary-layer, problem type. Although the initial, exponential grid input may give worse results than the power-law grid, using Powell's method to find the optimized, exponential grid produces the best results. The initial, control function, Q, must be chosen close to the final, optimized value. If it is chosen too far away from the final Q, Powell's method will not converge. Convergence is also affected by the number of iterations allowed and the spread between minimization guesses. If too, few iterations are done, the solution does not converge on an optimized value. If too many iterations are used, all of the convergence takes place in the initial steps of Powell's method. This wastes all subsequent calculations which check the convergence and is inefficient. Choosing a large spread between minimization guesses also is less efficient. Powell's method calculates a central, functional value, f, two functional values on either side of the central value, f^+ and f^-, and the minimum functional value of a quadratic fit through the previous three functional values, f^n. If the spread between the f, f^+, and f^- is too large, the solution

takes more iterations to converge. The successive iterations tend to continually overshoot the final value until getting very close. A large spread may be useful in problems where a good initial guess is not known, though. The function minimized also affects the efficiency of the adaptive-grid solution.

Powell's method can minimize any input function. This study optimizes the solution grid with the sum of $U_{\eta\eta\eta}{}^2$, $U_{\eta\eta}{}^2$, $U_{\eta}{}^2$, and of combinations of these three functions. Of the first three functions, minimizing the sum of $U_{\eta\eta\eta}{}^2$ best minimizes the error in U. It produces better results for C_f. The solution reduces the RMS of the error between the computed U and the von Karman-Pohlhausen approximation of U most. The computed velocity profile is also closest to Blasius', exact, velocity profile. Minimizing the sum of $U_{\eta}{}^2$ does not produce good results. The minimization does not converge on an optimum grid with this function. It does put more points into the boundary layer, but the results are worse for C_f and the velocity profile. In fact, minimizing $\sum U_{\eta}{}^2$, increases the error between the computed U and the von Karman-Pohlhausen solution. If $\sum U_{\eta}{}^2$ is included with any of the other derivative functions, the results are always worse. This is not the case with minimizing $\sum U_{\eta\eta}{}^2$. Minimizing with $\sum U_{\eta\eta}{}^2$ gets results very close to the results of minimizing the sum of $U_{\eta\eta\eta}{}^2$. Minimizing $\sum U_{\eta\eta}{}^2$ better resolves the flow at the leading edge, while minimizing $\sum U_{\eta\eta\eta}{}^2$ better resolves flow at the trailing edge. Consequently, the

combination of $\sum U_{\eta\eta}{}^2$ and $\sum U_{\eta\eta\eta}{}^2$ shows improved results over minimizing $\sum U^{\eta}{}_{\eta}{}^2$ separately. The $\sum U_{\eta\eta\eta}{}^2$ still produces the least error in U, however. Some other combination of the $\sum U_{\eta\eta}{}^2$ and the $\sum U_{\eta\eta\eta}{}^2$ may produce better results. Changing Reynold's number for the problem does not significantly change which minimized function produces the best results. Changing the number of grid points in the y direction seems to make the $\sum U_{\eta\eta}{}^2$ the best, minimizing function. For problems with smaller numbers of grid points, the affect of minimizing the $\sum U_{\eta\eta}{}^2$ increases. The number of grid points also affects the optimized Q value and the initial Q guess. Grids with more grid points converge on Q's closer to 0. Since the optimization inputs do affect the accuracy and behavior of the minimization process and final solution, the problem must be examined carefully to determine the inputs which give the best, adaptive-grid solution.

Using the adaptive-grid solutions in compressible, boundary-layer problems also produces good results. Computed velocity profiles are an excellent match to Van Driest's, theoretical solutions for high-speed flow over flat plates. The temperature, profile results are not quite as exact, but the computed, temperature profiles follow Van Driest's, theoretical profiles closely. The adaptive-grid results also compare well with high-speed Eckert theory predictions for the heat convection coefficient. Like the incompressible cases, minimizing the $\sum U_{\eta\eta\eta}{}^2$ produces the best results in compressible cases. Throughout the range from Mach 2 to

Mach 8, the computed h does not vary from Eckert's theoretical values by more than three percent. This is good correlation to an approximation like Eckert theory. Therefore, the adaptive grid solutions succeed in minimizing the error in U to give excellent velocity profiles for high speed, compressible flow and good agreement with high speed Eckert theory.

The boundary-layer code presented in this study is an improvement over previous methods. The non-adaptive, grid solutions are better than Lange's solutions and are close to Blasius,' exact, incompressible solution. Adaptive-grid solutions improve on the solutions using a non-adaptive grid and use fewer grid points. The adaptive grid also provides the flexibility to optimize the grid for any desired function and for general boundary-layer problems that may not have exact solution such as high-speed compressible or turbulent flow. The adaptive grid successfully solves compressible, flow problems. The application of the optimized, exponential, adaptive-grid method to these compressible problems shows that the method can be applied to more complicated problems than incompressible flow over a flat plate. Using a non-adaptive grid structured around a grid optimized for the flow solution, limits the application of the boundary-layer method. The adaptive grid has no such limitations. Further application of the adaptive-grid method could solve even more complicated problems. The solution of these problems may extend the engineer's knowledge of hypersonic, flow problems.

Chapter VII: Recommendations

This study has not investigated all possible applications or affects of the adaptive-grid optimization. Further studies of the optimization procedure could make some changes which could improve the efficiency of the optimization. Other changes to the boundary-layer code could also expand the applicability of the code to a much larger class of problems.

Two improvements to the optimization code could improve its efficiency. The biggest shortcoming in the adaptive-grid solution is that it requires much more computer time than a non-adaptive scheme since a solution of the boundary-layer code is required for each iteration of the method. Even though the number of grid points may be less, the number of iterations required for the adaptive-grid solution increases the overall, computer time. If a Thomas'-algorithm, iterative solution is used instead of the SOR, the time required to compute the boundary-layer solution decreases. Chen has included a quad-diagonal solver in this same, boundary-layer code to study turbulent, flow problems (5). This would dramatically reduce the computer time needed for adaptive-grid solutions. Also, minimizing some other combination of the sum of $U_{\eta\eta}{}^2$ and the sum of $U_{\eta\eta\eta}{}^2$ may produce better overall results than cases tested in this study. Other minimizing functions could also be tested. The parameters that form the

control function could also be changed. This study splits Q into two parts and minimizes each part. Q could be split into more parts, or a linear functional relationship between the parts could be used to form Q. Many different parameters can be varied to find the best optimization method. The flexibility of Powell's method provides many avenues which might result in better answers for various problems.

There are additions that can be made to apply the optimization grid to a larger class of problems. To further reduce the error in the boundary layer code and make it a true, two-dimensional problem, the adaptive grid could be applied to both streamwise and normal directions. This study only optimized in the normal direction and then scaled the exponential stretching in the streamwise direction. Turbulence models, and nonisothermal wall effects could be inserted. The axisymmetric, boundary-layer equations could be used to solve flows over axisymmetric bodies. This would allow the solution of a wide variety of boundary-layer flows with the adaptive grid. The possibilities for using the basic optimization method presented in this study are limitless. But much more study is required to determine the application of adaptive-grid and boundary-layer solutions to investigations of hypersonic, flow regimes.

Bibliography

1. Acton, Forman S. Numerical Methods That Work, New York: Harper and Row, 1970.

2. Anderson, D. A., J. C. Tannehill, and R. H. Plecher. Computational Fluid Mechanics and Heat Transfer. New York:McGraw-Hill Book Company, 1984.

3. Boyd, Bruce D. Adaptive Grid Generation for Numerical Solution of Burger's Equation, M.S. Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1983.

4. Cappelano, P. T. Heat Transfer/Boundary Layer Investigation of Heating Discrepancies in Wind Tunnel Testing of Orbiter Insulating Articles, M.S. Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1982.

5. Chen, Alice J., research for M.S. Thesis to be submitted March 1986, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio.

6. Fiore, A. W.. "Viscosity of Air," Journal of Spacecraft and Rockets, 3 (5): 56-58 (May 1966).

7. Ghia, K. N., U. Ghia, and C. T. Shin. "Adaptive Grid Generation for Flows with Local High Gradient Regions," Advances in Grid Generation, FED-Vol. 5, AMSE, June 20,22, pp.35-47, (1983).

8. Hodge, James K., Associate Professor of Aeronautics, Air Force Institute of Technology, October 1985.

9. Hodge, James K., Sal A. Leone, and McCarty, R.L., "Non-Iterative Parabolic Grid Generation for Parabolized Equations," presented at the AIAA 7th Computational Fluid Dynamics Conference, AIAA Paper 85-1528-CP, 1985.

10. Hodge, J.K. and A.L. Stone, "Numerical Solutions for Airfoils Near Stall in Optimized Boundary Fitted Curvilinear Coordinates," AIAA Paper 78-284,AIAA 16th Aerospace Schences Meeting, Huntsville, Alabama, January 16-18, 1978.

11. Hodge, James K., Y. K. Woo, and P. T. Cappelano. "Parameter Estimation for Imbedded Thermocouples in Space Shuttle Wind Tunnel Test Articles with a Nonisothermal Wall," AIAA 83-1533, June 1983.

12. Holman, J. P. Heat Transfer, New York: McGraw-Hill, Inc., 1981.

13. Keyes, F.G., "A Summary of Viscosity and Heat Conduction Data for He, A, H_2, N_2, CO, H_2O, and Air," Transactions of ASME, 73:589-596, (1951).

14. Kuethe, A. M. and A. Y. Chow. Foundations of Aerodynamics. New York, John Wiley and Sons, Inc., 1976.

15. Lange, Karen J.. Unsteady Solution of the Boundary Layer Equations with Application to Space Shuttle Tiles. M.S. Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, May 1985.

16. Leone, Sal A. and James K. Hodge. "Minimizing the Steady State Truncation Error," Society of Industrial and Applied Mathematics Fall Meeting, Arizona State University, Tempe, Arizona, October 1985.

17. Liepmann, H.W. and A. Roschko. Elements of Gasdynamics. New York: John Wiley and Sons,Inc., 1957.

18. Nagamatsu, H.T. and R.E. Sheer,Jr. "Hypersonic Shock Wave Boundary Layer Interaction and Leading Edge Slip," ARS Journal, pp. 454 -462. May 1960.

19. Powell, M.J.D. "An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives." Computer Journal, Vol. 7, pp.155-162, 1964.

20. Rasmussen, M. L. Lecture Notes distributed in AE 630, Reentry Aerodynamics. School of Engineering, Air Force Instutute of Technology (AU), Wright-Patterson AFB,OH, April 1985.

21. Roberts, Timothy K. A Numerical Solution of a Nonisothermal Wall Using the Two-Dimensional Navier-Stokes Equations, M.S. Thesis, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1984.

22. Schetz, Joseph A.. Foundations of Boundary Layer Theory, Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1984.

23. Schlichting, H.. Boundary Layer Theory. New York: McGraw-Hill Book Company, 1979.

24. Van Driest, E. R.. "Investigation of Laminar Boundary Layer in Compresible Fluids Using the Crocco Method," NACA Technical Note, 2597: Washington (Jan 1952).

Appendix A: Compressibility Transformation

The boundary-layer equations under most conditions are density dependent. Compressiblity makes the solution of the non-linear, boundary-layer equations more difficult. If the equations are transformed into equations similar to the incompressible, boundary-layer equations, a solution might be easier. To transform the compressible equations, some form of a compressibility transformation is used. The transformation puts the density dependence into the varibles of the equations. In the physcial space, the dimensional variables are x', y', and t'. The variables are non-dimensionalized using

$$x = \frac{x'}{L'}, \quad y = \frac{y'}{L'}, \quad t = \frac{t'U'_\infty}{L'}, \quad u = \frac{u'}{U'_\infty}, \quad v = \frac{v'}{U'_\infty}, \tag{A-1}$$

$$H = \frac{H'}{U'^2_\infty}, \quad T = \frac{C_p'T'}{U'^2_\infty}, \quad p = \frac{p'}{\rho_\infty' U'^2_\infty}, \quad \mu = \frac{\mu'}{\mu_\infty'}, \quad \rho = \frac{\rho'}{\rho_\infty'},$$

The transformed variables are

$$X = x, \qquad Y = \int_0^y \frac{\rho'}{\rho'_e}\, dy, \qquad t' = t \tag{A-2}$$

The non-dimensional, compressible, axisymmetric boundary-layer equations are

Continuity:
$$r^m \frac{\partial \rho}{\partial t} + \frac{\partial r^m \rho u}{\partial x} + \frac{\partial r^m \rho v}{\partial y} = 0 \tag{A-3}$$

Momentum:
$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2}{\partial x} + \frac{\partial \rho u v}{\partial y} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial y}\left(\frac{\mu}{Re}\frac{\partial u}{\partial y}\right) \tag{A-4a}$$

Momentum: $$\frac{\partial p}{\partial y} = 0 \qquad \text{(A-4b)}$$

Energy: $$\frac{\partial \rho H}{\partial t} + \frac{\partial \rho u H}{\partial x} + \frac{\partial \rho v H}{\partial y} = \frac{\partial p}{\partial t} + \frac{\partial}{\partial y}\left(\frac{\mu}{RePr}\frac{\partial H}{\partial y}\right) + \frac{\partial}{\partial y}\left(\frac{\mu(Pr-1)}{RePr}\frac{\partial u^2/2}{\partial y}\right) \qquad \text{(A-5)}$$

where m =0 for two-dimensional problems and m=1 for axisymmetric problems (Appendix B). The transformed variables must be substituted into the compressible, boundary-layer equations. To substitute for the partial derivative terms use the following relationships:

$$\frac{\partial W}{\partial x} = \frac{\partial W}{\partial X}\frac{\partial X}{\partial x} + \frac{\partial W}{\partial Y}\frac{\partial Y}{\partial x}, \quad \frac{\partial W}{\partial y} = \frac{\partial W}{\partial X}\frac{\partial X}{\partial y} + \frac{\partial W}{\partial Y}\frac{\partial Y}{\partial y}, \quad \frac{\partial W}{\partial t} = \frac{\partial W}{\partial t} \qquad \text{(A-6)}$$

where $\frac{\partial X}{\partial t} = \frac{\partial Y}{\partial t} = \frac{\partial t}{\partial x} = \frac{\partial t}{\partial y} = \frac{\partial X}{\partial y} = 0$ and $\frac{\partial Y}{\partial y} = \rho$

W in the above equations represents any variable of interest. After expanding Eq (A-3) using the chain rule and then substituting relations from Eq (A-6), the transformed, continuity equation becomes

$$r^m\frac{\partial \rho}{\partial t} + r^m\frac{\partial \rho}{\partial Y}\frac{\partial Y}{\partial t} + ur^m\frac{\partial \rho}{\partial x} + \rho r^m\left(\frac{\partial U}{\partial X} + \frac{\partial U}{\partial Y}\frac{\partial Y}{\partial x}\right) + r^m v\left(\frac{\partial \rho}{\partial Y}\frac{\partial Y}{\partial y}\right) + \rho r^m\frac{\partial V}{\partial Y}\frac{\partial Y}{\partial y} = 0 \qquad \text{(A-7)}$$

The desired form of the incompressible, axisymmetric, continuity equation is

$$\frac{\partial r^m U}{\partial X} + \frac{\partial r^m V}{\partial Y} = 0 \qquad \text{(A-8)}$$

The continuity equation in axisymmetric, physical space coordinates is

$$\frac{\partial r^m u}{\partial x} + \frac{\partial r^m v}{\partial y} = 0 \qquad \text{(A-9)}$$

Compare Eqs (A-8) and (A-9) to determine what U is.

$$\frac{\partial r^m}{\partial x} = r^m \frac{\partial u}{\partial X}\frac{\partial X}{\partial x} = r^m \frac{\partial u}{\partial X} = r^m \frac{\partial U}{\partial X} \qquad \text{(A-10)}$$

Therefore,

$$U = u \qquad \text{(A-11)}$$

Taking out the $\partial U/\partial X$ term in Eq (A-7), the remaining terms are the second term in Eq (A-8).

$$r^m \frac{\partial \rho}{\partial t} + u r^m \frac{\partial \rho}{\partial x} + \rho r^m \frac{\partial U}{\partial Y}\frac{\partial Y}{\partial x} + \frac{\partial r^m \rho v}{\partial y} = \frac{\partial r^m v}{\partial y} = \rho r^m \frac{\partial V}{\partial Y}$$

where $\rho = \partial Y/\partial y$ (A-12)

$$r^m \frac{\partial^2 Y}{\partial y \partial t} + r^m u \frac{\partial^2 Y}{\partial y \partial x} + r^m \frac{\partial u}{\partial y}\frac{\partial Y}{\partial x} + \frac{\partial r^m \rho v}{\partial y} = \rho r^m \frac{\partial V}{\partial y} \qquad \text{(A-13)}$$

and

$$r^m \rho v = r^m V - r^m \partial Y/\partial t - u r^m \partial Y/\partial x \qquad \text{(A-14)}$$

Therefore,

$$V = \rho v + \partial Y/\partial t + u \partial Y/\partial x \qquad \text{(A-15)}$$

The transformed velocity components, U and V, satisfy Eq (A-8). When they are substituted into Eqs (A-4) and (A-5), equations similar to the incompressible, momentum and energy equations result (15:81-83). These equations are

Momentum:

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial X} + V\frac{\partial U}{\partial Y} = -\frac{1}{\rho}\frac{\partial p}{\partial X} + \frac{\partial}{\partial Y}\left(\frac{\rho \mu}{Re}\frac{\partial U}{\partial Y}\right) \qquad \text{(A-16a)}$$

$$\frac{\partial p}{\partial Y} = 0 \qquad \text{(A-16b)}$$

Energy:

$$\frac{\partial H}{\partial t} + U\frac{\partial H}{\partial X} + V\frac{\partial H}{\partial Y} = \frac{1}{\rho}\frac{\partial p}{\partial t} + \frac{\partial}{\partial Y}\left(\frac{\rho \mu}{RePr}\frac{\partial H}{\partial Y}\right) + \frac{\partial}{\partial Y}\left(\frac{\rho \mu (Pr-1)}{RePr}\frac{\partial U^2/2}{\partial Y}\right) \qquad \text{(A-17)}$$

Appendix B: Axisymmetric, Boundary-Layer Equations

The two-dimensional analysis of boundary-layer equations is easily applied to axisymmetric shapes. Axisymmetric shapes have one axis such that a plane passed perpendicular through this axis slices out a circle of varying radius. Fig. B-1 shows a general, axisymmetic, coordinate system. Any location on the surface is described by coordinates x, y, and ϕ. The scale factors are h_i. Velocity components are V_i. h_i and V_i are

$$\begin{aligned} h_1 &= 1 + y/R_1 & V_1 &= u \\ h_2 &= 1 & V_2 &= v \\ h_3 &= R_o(x) + y\cos\alpha & V_3 &= 0 \end{aligned} \qquad \text{(B-1)}$$

R_o is the local radius of the surface and varies with x. R_1 is the local radius of curvature. For a cone, α is 0^o, and there is no x-dependence in y. The only non-unity, scale factor is h_3 which becomes $R_o(x)$, or just r. To make the two-dimensional equations more applicable to two-dimensional and conical, axisymmetric problems, the superscipt m is added. For two-dimensional problems, m is 0. For axisymmetric problems, m is 1. In this study, the two-dimensional, incompressible, boundary-layer equations were transformed into a computational plane where

$$\begin{aligned} \xi &= \xi(X, Y, t) \\ \eta &= \eta(X, Y, t) \\ \tau &= t \end{aligned} \qquad \text{(B-2)}$$

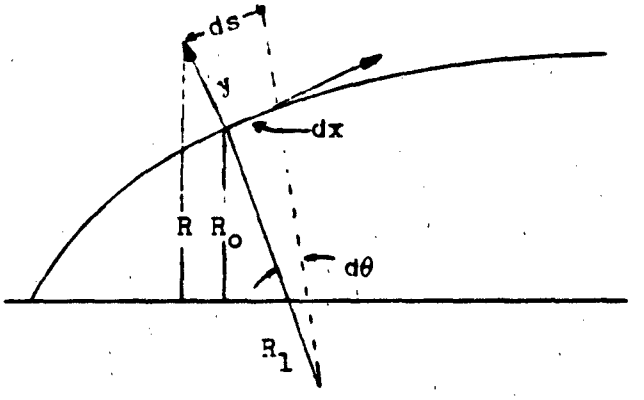


Fig. B-1 General, Axisymmetric Coordinates

After the metric relationships are applied, the transformed, two-dimensional, incompressible, boundary-layer equations are

$$\text{Continuity:} \quad \frac{\partial r^m U}{\partial \xi}\,\xi_X + \frac{\partial r^m U}{\partial \xi}\,\eta_X + \frac{\partial r^m V}{\partial \xi}\,\eta_Y = 0 \tag{B-3}$$

$$\text{Momentum:} \quad U_t + U(U_\xi \xi_X + U_\eta \eta_X) + VU_\eta \eta_Y = 1/\mathrm{Re}[\mu\rho(U_\eta \eta_Y)]\eta_Y \tag{B-4}$$

$$\text{Energy:} \quad H_t + U(H_\xi \xi_X + H_\eta \eta_X) + VH_\eta \eta_Y = \tag{B-5}$$

$$p_t/\rho + \left(\frac{\rho\mu}{\mathrm{PrRe}} H_\eta \eta_Y\right)\eta_Y + \left[\frac{\rho\mu(\mathrm{Pr}-1)}{2\mathrm{RePr}}\,\eta_Y \left(\frac{U^2}{2}\right)_\eta\right]\eta_Y$$

The momentum and energy equations are not affected by the changing radius of a cone's surface. Consequently, the finite-difference equations for finding U and H do not change. The solution of V requires integrating the continuity equation which does depend on the changing radius.

The continuity equation is rearranged to isolate V. The resulting eqaution is then integrated to solve for V. The integrable equation is

$$(r^m V)_\eta = \frac{-Y_\xi}{X_\xi}(r^m U)_\xi + \frac{Y_\xi}{X_\xi}(r^m U)_\eta \tag{B-6}$$

The right-hand, side term with $(r^m U)_\eta$ is differenced with a three-point, windward scheme about the j+1/2 point. The second term splits up into parts.

$$\frac{Y_\xi}{X_\xi}(r^m U)_\eta = \frac{1}{X_\xi}\left[\, Y_\xi (r^m U) - \int (r^m U)(Y_\xi)_\eta \, d\eta \,\right]\Bigg|_{j-1}^{j} \tag{B-7}$$

The integral is evaluated using a trapezoidal rule with Y averaged about the j-1/2 point prior to the integration.

$$Y_\xi/X_\xi(r^m U)_\eta = 1/X_\xi \{[(Y_\xi r^m U)_{i,j}-(Y_\xi r^m U)_{i,j-1}] \quad \text{(B-8)}$$
$$- .5[(r^m U)_{i,j}+(r^m U)_{i,j-1}][Y_{\xi i,j} - Y_{\xi i,j-1}]\}$$

$$= 1/(2X_\xi) [(Y_\xi r^m U)_{i,j}-(Y_\xi r^m U)_{i,j-1} \quad \text{(B-9)}$$
$$+ Y_{\xi i,j-1}(r^m U)_{i,j}-Y_{\xi i,j}(r^m U)_{i,j-1}]$$

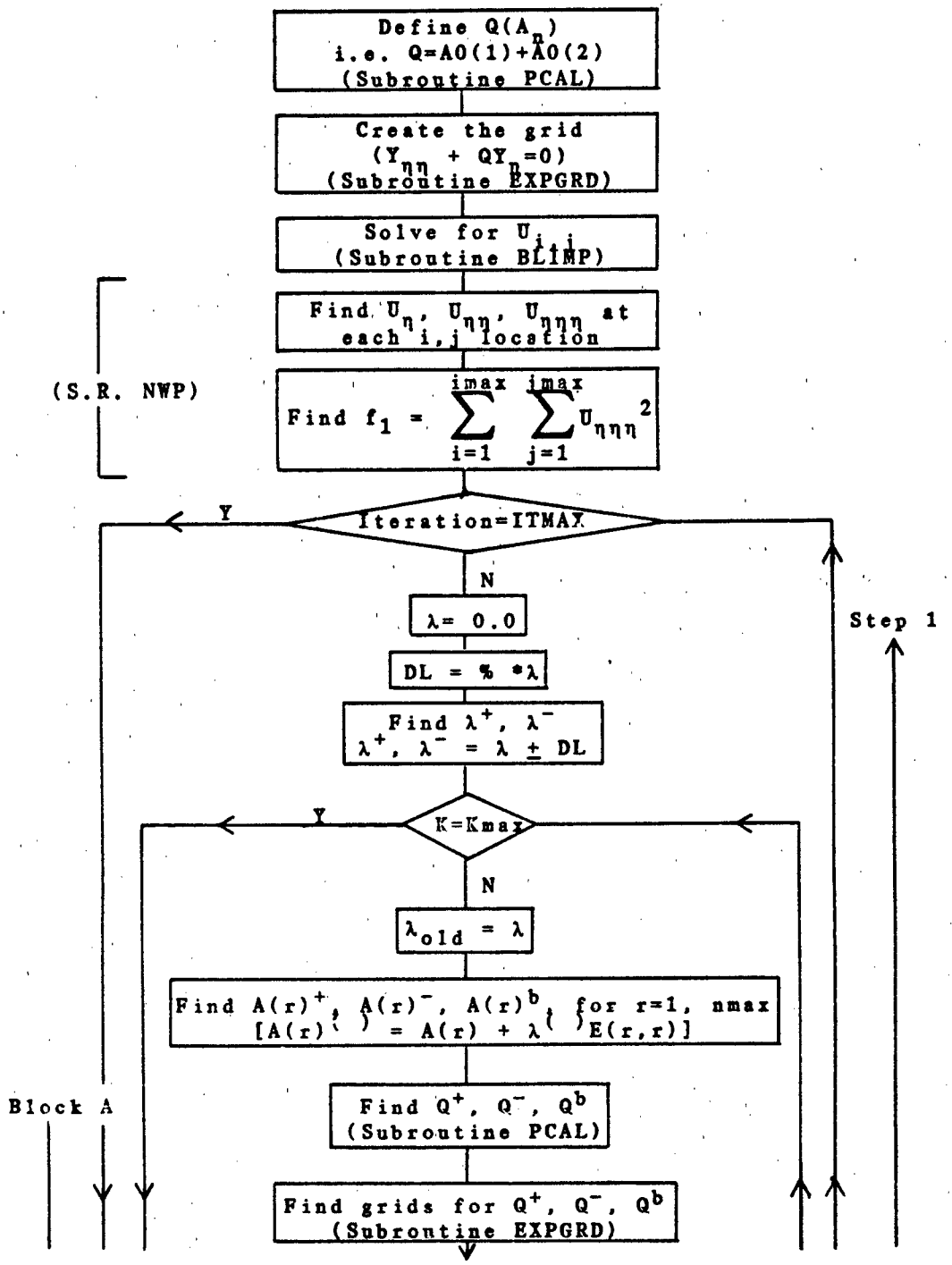
$$= 1/2X_\xi\{(Y_{\xi i,j}+Y_{\xi i,j-1})[(r^m U)_{i,j}-(r^m U)_{i,j-1}]\} \quad \text{(B-10)}$$

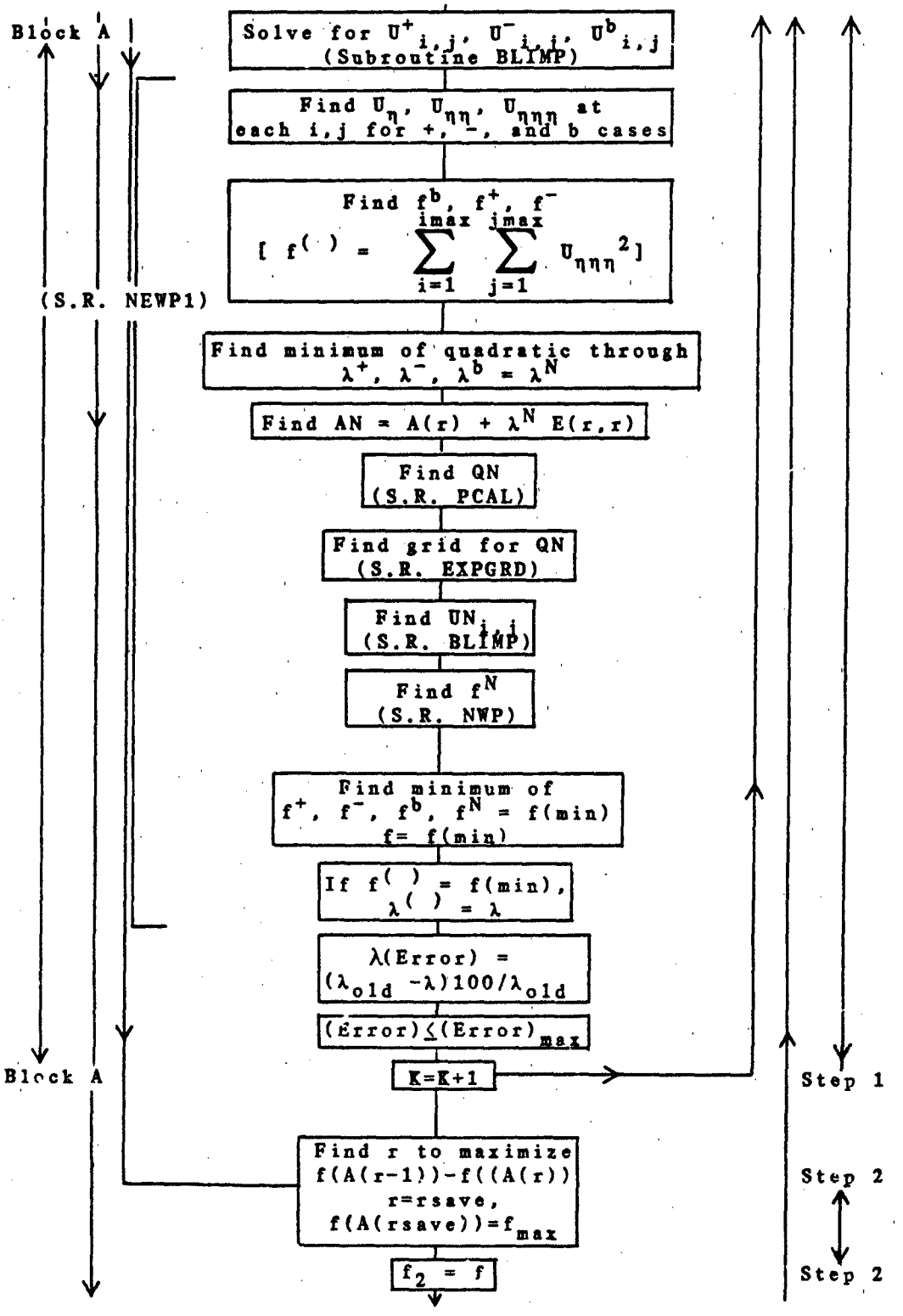
With the other terms of V_η added in, the finite difference for $V_{i,j}$ is

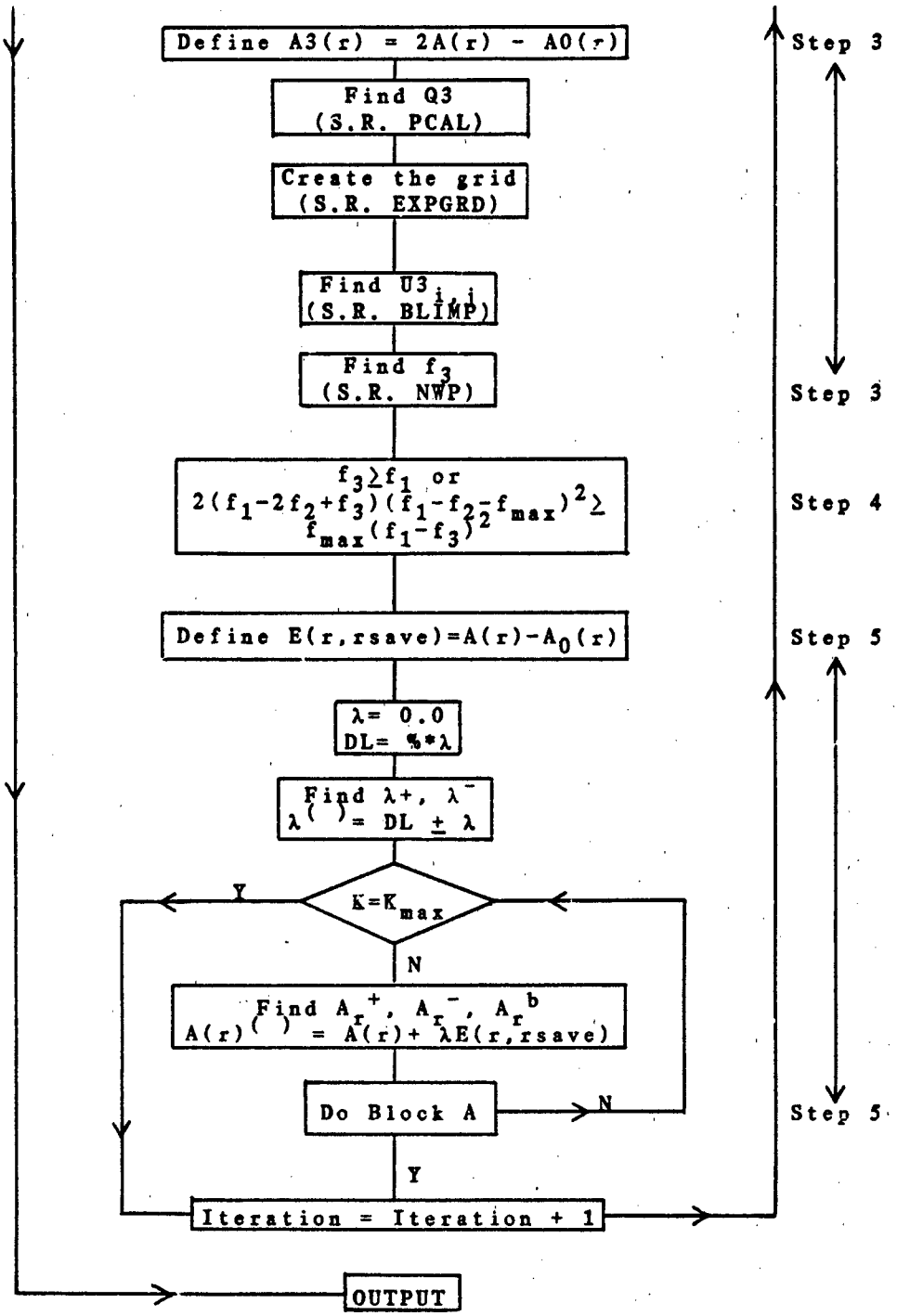
$$V_{i,j} = 1/(r^m)_{i,j} \{(r^m V)_{i,j-1}$$
$$+ 1/(2X_\xi)\{.5(Y_{i,j}-Y_{i,j-1})\{-3[(r^m U)_{i,j}+(r^m U)_{i,j-1}]$$
$$+4[(r^m U)_{i-1,j}+(r^m U)_{i-1,j-1}]-[(r^m U)_{i-2,j}+(r^m U)_{i-2,j-1}]\}$$
$$+ (Y_{\xi i,j}+Y_{\xi i,j-1})[(r^m U)_{i,j}-(r^m U)_{i,j-1}]\}\} \quad \text{(B-11)}$$

This finite difference is then iterated to solve for V at each location in the domain.

Appendix C: Optimization Flow Chart




Block A
(S.R. NEWP1)
Block A
Solve for $U^+_{i,j}$, $U^-_{i,j}$, $U^b_{i,j}$ (Subroutine BLIMP)
Find U_η, $U_{\eta\eta}$, $U_{\eta\eta\eta}$ at each i,j for +, −, and b cases
Find f^b, f^+, f^- $[f^{(\ )} = \sum_{i=1}^{imax}\sum_{j=1}^{jmax} U_{\eta\eta\eta}{}^2]$
Find minimum of quadratic through λ^+, λ^-, $\lambda^b = \lambda^N$
Find AN = A(r) + λ^N E(r,r)
Find QN (S.R. PCAL)
Find grid for QN (S.R. EXPGRD)
Find $UN_{i,j}$ (S.R. BLIMP)
Find f^N (S.R. NWP)
Find minimum of f^+, f^-, f^b, f^N = f(min) f= f(min)
If $f^{(\ )}$ = f(min), $\lambda^{(\ )} = \lambda$
λ(Error) = $(\lambda_{old} - \lambda)100/\lambda_{old}$
(Error) ≤ $(Error)_{max}$
K=K+1
Find r to maximize f(A(r-1))-f((A(r)) r=rsave, f(A(rsave))=f_{max}
f_2 = f
Step 1
Step 2
Step 2


Define A3(r) = 2A(r) - A0(r)
Find Q3 (S.R. PCAL)
Create the grid (S.R. EXPGRD)
Find U3 i,j (S.R. BLIMP)
Find f3 (S.R. NWP)
f3≥f1 or 2(f1-2f2+f3)(f1-f2-fmax)^2≥ fmax(f1-f3)^2
Define E(r,rsave)=A(r)-A0(r)
λ= 0.0
DL= %*λ
Find λ+, λ-
λ( )= DL ± λ
K=Kmax
Y
N
Find Ar+, Ar-, Arb
A(r)( ) = A(r)+ λE(r,rsave)
Do Block A
N
Y
Iteration = Iteration + 1
OUTPUT
Step 3
Step 3
Step 4
Step 5
Step 5

Appendix D: Optimization Program

```
      PROGRAM MNTRER
C
C  THIS PROGRAM USES POWELL'S METHOD OF CONJUGATE DIRECTIONS
C  TO MIN THE TRUNCATION ERROR IN A BOUNDARY LAYER
C  THIS PROGRAM WILL COMPUTE THE FOLLOWING:
C  NUMERICAL GRID SOLUTION USING POWELL'S METHOD AND
C  THE CONTROL FUNCTION P TO MINIMIZE TRUNCATION ERROR
C  IN THE TRANSFORMED ETA DIRECTION OF A BOUNDARY LAYER
C  GRID
C  IN ALL CASES THE THIRD DERIVATIVE (DUUU) IS CALCULATED NUMERICALLY
C
C
      DIMENSION UEXACT(21,0:20),P(19),X(21),DUUU(21)
      DIMENSION Y(21,19),YPLUSDL(21,19),YMINDL(21,19)
      DIMENSION P3(19),D3U(21),Y3(21,19),U3(21,0:20)
      DIMENSION AAA(19),BBB(19),CCC(19),DDD(19),GGG(19),WWW(19)
      DIMENSION PPLUSDL(19),PMINDL(19)
      DIMENSION U(21,0:20),UPLUSDL(21,0:20),UMINDL(21,0:20)
      DIMENSION A(5),ASAVE(5),A3(5),APLUSDL(5),AMINDL(5)
      DIMENSION ABAR(5),ZI(2,2),DELTA(21),ABAR2(5)
      DIMENSION POLD(19)
      OPEN(UNIT=8,FILE='RESTART')
      OPEN(UNIT=9,FILE='WALLQ')
      OPEN(UNIT=10,FILE='FIELD')
      OPEN(UNIT=11,FILE='GRID')
      OPEN(UNIT=12,FILE='HREF')
      OPEN(UNIT=13,FILE='CFCF')
      REWIND 6
      REWIND 8
      REWIND 9
      REWIND 10
      REWIND 11
      REWIND 12
      REWIND 13
C
      CALL SECOND(CPI)
      READ (7,*) IMAX,JMAX,KMAX,ERMXL
      READ (7,*) YMIN,YMAX,PERCENT
      READ (7,*) NMAX,ITMAX
      READ (7,*) ERPOWMX,ICOL,IRST
      READ (7,*) (A(IR),IR=1,NMAX)
      READ (7,*) WHO,WH2,WH3
      READ (7,*) Z
      CALL DATE(ADATE)
      CALL TIME(ATIME)
      WRITE(6,100) ADATE,ATIME
  100 FORMAT('1',/,5X,A10,2X,A10)
      WRITE(6,200) IMAX,JMAX,YMIN,YMAX,KMAX,ERMXL,PERCENT,
     1             (A(IR),IR=1,NMAX)
  200 FORMAT(1X,'NUMBER OF GRID PTS, IMAX=',I3,3X,'JMAX=',
     1       I3,/,1X,'MIN Y VALUE, YMIN=',E13.5,/,1X,
     2       'MAX Y VALUE, YMAX=',E13.5,/,1X,
     3       'MAX NUMBER OF ITERATIONS FOR ONE',
     4       'PARAMETER OPTIMIZATION, KMAX=',I5,/,1X,
     5       'MAX PERCENT ERROR IN ONE PARAMETER',
     6       ' OPTIMIZATION FOR CONVERGENCE, ERMXL=',
     7       E13.5,/,1X,'PERCENT CHANGE IN PARAMETERS USED',
     8       'TO CALCULATE QUAD EQ FOR ONE PARAMETER OPT=',
     9       E13.5,/,1X,'INITIAL A ',5E13.5,/,/)
      WRITE (6,209) WHO,WH2,WH3
  209 FORMAT (1X,'THE WEIGHTING FOR DU = ',E7.2,2X,
     1        'DUU =',E7.2,2X,'DUUU =',E7.2,/)
      WRITE (6,211) ITMAX
  211 FORMAT(1X,'MAX NUMBER OF ITERATIONS,IN OUTER LOOP',
```

```
     1       'TO POWELL METHOD, ITMAX=',I5)
        WRITE(6,213)ICOL
 213    FORMAT(1X,'OPTIMIZED COLUMN IS',1X,I3)
C
       PI=ACOS(-1.)
       JMM1=JMAX-1.
C
C  SETTING UP BOUNDARY CONDITIONS
       XSTEP=1/(FLOAT(IMAX)-1.)
       DO 5 I=1,IMAX
        X(I)=FLOAT(I-1)*XSTEP
 5     CONTINUE
       Y(ICOL,1)=YMIN
       Y(ICOL,JMAX)=YMAX
C
C  THIS SECTION CALCULATES THE OPTIMIZED NUMERICAL SOLUTION
C  USING POWELL'S METHOD (1964)
C
        DO 50 J=1,NMAX
         ASAVE(J)=A(J)
         DO 70 IR=1,NMAX
          ZI(J,IR)=0.0
 70      CONTINUE
         ZI(J,J)=A(J)*0.1
         IF (ZI(J,J).EQ.0.0) ZI(J,J)=.01
 50     CONTINUE
        IB=0
        CALL PCAL(A,NMAX,P,JMAX)
 55     CALL EXPGRD(X,Y,P,AAA,CCC,DDD,GGG,WWW,YMIN,
     1                  YMAX,IMAX,JMAX,ICOL)
        CALL BLIMP(X,Y,IMAX,JMAX,U,IB,UE,DELTA,RE)
         WRITE(6,*) 'DELTA= ',DELTA(IMAX)
       IF (YMAX.LT.2*DELTA(IMAX)) THEN
        YMAX=DELTA(IMAX)*2
        Y(ICOL,JMAX)=YMAX
        GO TO 55
       ENDIF
        CALL NWP (U,DUUU,H1,IMAX,JMAX,ICOL,WH0,WH2,WH3)
        H=H1
        CALL POHL(Y,DELTA,UE,IMAX,JMAX,UEXACT)
         WRITE (6,402)
 402    FORMAT (3X,'I',4X,'J',09X,'P',12X,'X',12X,'Y',10X,
     1          'UEXACT',8X,'U',11X,'U-UEXACT')
        DO 611 I=1,IMAX
         IF(I.EQ.2.OR.I.EQ.IMAX) THEN
         DO 612 J=1,JMAX
          WRITE (6,502) I,J,P(J),X(I),Y(I,J),UEXACT(I,J),
     1          U(I,J),U(I,J)-UEXACT(I,J)
 502      FORMAT (1X,2I3,6E13.5)
 612     CONTINUE
         ENDIF
 611    CONTINUE
        WRITE (6,602)
 602   FORMAT (1X,' P IS JUST A FUNCTION OF THE INITIAL A',
     1          'VALUES AND NOT OF RLAMBA OR ZI',/)
        CALL ERROR(U,UEXACT,IMAX,JMAX)
        ITRT=1
        IF (ITRT.GT.NMAX) GO TO 580
        DO 370 ITERATE=ITRT,ITMAX
           HOLD=H
           IRSAVE=0
           HDELMAX=0.0
           ERRORPW=0.0
          DO 150 IR=1,NMAX
           ERRORPW=ERRORPW+ABS(A(IR)-ASAVE(IR))
           ASAVE(IR)=A(IR)
 150      CONTINUE
          ERRORPW=ERRORPW/NMAX
          IF (ERRORPW.LE.ERPOWMX.AND.ITERATE.NE.1) GO TO 580
          WRITE (6,122) ITERATE
```

```
 122      FORMAT (1X,/,/,/,'.....BELOW ITERATE=',I5,
     1          '.....')
      IF(IRST.EQ.1) THEN
C
C  STEP 1 OF POWELL'S ALGORITHM:  FOR IR=1,2,...NMAX CALCULATE RLAMBA
C  SO THAT H(A+RLAMBA*ZI) IS A MINIMUM AND DEFINE A(IR)=A(IR-1)+RLAMBA*ZI
C
      IR1=1
      DO 160 IR=IR1,NMAX
         RLAMBA=0.0
         DLTA=PERCENT*RLAMBA
         IF (ABS(DLTA).LE..05) DLTA=.05
         RLAMPL=RLAMBA+DLTA
         RLAMMN=RLAMBA-DLTA
C
C
C  ITERATION FOR ONE PARAMETER OPTIMIZATION
C
              DO 250 K=1,KMAX
               RLAMOLD=RLAMBA
               DO 180 J=1,NMAX
                APLUSDL(J)=A(J)+RLAMPL*ZI(J,IR)
                AMINDL(J)=A(J)+RLAMMN*ZI(J,IR)
                ABAR(J)=A(J)+RLAMBA*ZI(J,IR)
 180           CONTINUE
               CALL PCAL(APLUSDL,NMAX,PPLUSDL,JMAX)
               CALL PCAL(AMINDL,NMAX,PMINDL,JMAX)
               CALL PCAL(ABAR,NMAX,P,JMAX)
             CALL EXPGRD(X,Y,P,AAA,CCC,DDD,GGG,WWW,YMIN,
     1                   YMAX,IMAX,JMAX,ICOL)
            CALL EXPGRD (X,YPLUSDL,PPLUSDL,AAA,CCC,DDD,GGG,WWW,YMIN,YMAX,
     1                   IMAX,JMAX,ICOL)
             CALL EXPGRD (X,YMINDL,PMINDL,AAA,CCC,DDD,GGG,WWW,YMIN,YMAX,
     1                   IMAX,JMAX,ICOL)
C
      CALL BLIMP(X,Y,IMAX,JMAX,U,ITERATE,UE,DELTA,RE)
      CALL BLIMP(X,YPLUSDL,IMAX,JMAX,UPLUSDL,ITERATE,UE,DELTA,RE)
      CALL BLIMP(X,YMINDL,IMAX,JMAX,UMINDL,ITERATE,UE,DELTA,RE)
       CALL NEWP1 (U,P,UPLUSDL,UMINDL,DUUU,RLAMPL,RLAMMN,
     1             RLAMBA,RMSDUUU,H,K,PERCENT,IMAX,JMAX,
     2  ABAR2,NMAX,X,Y,AAA,CCC,DDD,GGG,WWW,YMIN,YMAX,ICOL,
     3 ITERATE,UE,DELTA,P3,Y3,D3U,U3,A,ZI,IR,WHO,WH2,WH3)
        WRITE(6,800) IR,(A(J),J=1,NMAX)
 800   FORMAT(1X,'ABOVE RESULTS ARE FOR INNER OPT AND',
     1        'A(IR) PARAMETER, IR=',I5,/,' THE OLD A(I)',
     2         'VALUES ARE=',5E13.5)
       WRITE (6,801) (ZI(J,IR),J=1,NMAX)
 801    FORMAT (1X,' THE OLD ZI(J,IR) VALUES=',5E13.5,/,
     1          1X,' POLD IS CALCULATED BY USING',
     2          ' A=A(OLD)+(RLAMBA OLD)*(ZI OLD)',/,/)
        ERRORL=ABS(RLAMOLD-RLAMBA)*100.0
          IF (RLAMOLD.NE.0.0) ERRORL=ERRORL/ABS(RLAMOLD)
            IF (ERRORL.LT.ERMXL) THEN
             WRITE(6,*) 'K-CONVERGED'
             GO TO 290
            ENDIF
 250        CONTINUE
 290        CONTINUE
              DO 310 J=1,NMAX
               A(J)=A(J)+RLAMBA*ZI(J,IR)
 310           CONTINUE
C
C  STEP 2:  FIND THE INTEGER IRSAVE SO THAT H(A(IR-1))-
C  H(A(IR)) IS A MAXIMUM, AND DEFINE THE MAXIMUM AS HDELTMAX
C
               HDELTA=HOLD-H
               HOLD=H
                 IF(HDELTA.GT.HDELMAX) THEN
                 HDELMAX=HDELTA
                 IRSAVE=IR
```

```
                END IF
 160            CONTINUE
C
C  STEP 3:  CALCULATE H3=H(2*A(NMAX)-A(O)), AND DEFINE H1=H(A(O)),
C           AND H2=H(A(NMAX))
C
 163      H2=H
         IRST=0
                DO 320 IR=1,NMAX
                 A3(IR)=2*A(IR)-ASAVE(IR)
 320            CONTINUE
       CALL PCAL(A3,NMAX,P3,JMAX)
       CALL EXPGRD(X,Y3,P3,AAA,CCC,DDD,GGG,WWW,YMIN,YMAX,IMAX,JMAX,ICOL)
       CALL BLIMP(X,Y3,IMAX,JMAX,U3,ITERATE,UE,DELTA,RE)
WRITE(6,*) 'THIS NWP CALLED IN STEP 3'
       CALL NWP(U3,D3U,H3,IMAX,JMAX,ICOL,WHO,WH2,WH3) C
C STEP 4:  IF EITHER H3.GE.H1 AND/OR
C     (2*(H1-2*H2+H3)*(H1-H2-HDELMAX)**2).GE.(HDELMAX*(H1-H3)**2)
C     USE THE OLD DIRECTIONS ZI FOR THE NEXT ITERATION AND
C     USE A(NMAX) FOR THE NEXT A(O) (GO TO STEP 1), OTHERWISE
C           GO TO STEP 5
C
          RLHS=2.*(H1-2.*H2+H3)*(H1-H2-HDELMAX)**2
          RHS=HDELMAX*(H1-H3)**2
          IF (H3.GE.H1.OR.RLHS.GE.RHS) GO TO 550
C
C STEP 5:DEFINE ZI=A(NMAX)-A(O) CALCULATE RLAMBA SO THAT
C        H(A(NMAX)+RLAMBA*ZI) IS A MINIMUM AND REPLACE THE
C        ZI(IRSAVE) VALUE BY THE ZI VALUE ABOVE AND USE
C        A(NMAX)+RLAMBA*ZI AS THE STARTING POINT FOR THE NEXT
CC       ITERATION
C
       IR1=1
 391   IF(IRST.EQ.1) THEN
       READ(8,*)IR1,HDELMAX,IRSAVE,((ZI(J,I),J=1,NMAX),I=1,NMAX)
       IRST=0
       ENDIF
       IF(IR1.GT.NMAX) GO TO 392
        DO 390 IR=IR1,NMAX
          ZI(IR,IRSAVE)=A(IR)-ASAVE(IR)
          IF (ZI(IR,IRSAVE).EQ.0.0) ZI(IR,IRSAVE)=.01
 390    CONTINUE
 392    CONTINUE
          RLAMBA=0.0
          DLTA=PERCENT*RLAMBA
          IF (ABS(DLTA).LE..05) DLTA=.05
           RLAMPL=RLAMBA+DLTA
           RLAMMN=RLAMBA-DLTA
C
C ONE PARAMETER OPTIMIZATION FOR STEP 5 OF ALGORITHM
C
           DO 480 K=1,KMAX
            RLAMOLD=RLAMBA
            DO 440 J=1,NMAX
             APLUSDL(J)=A(J)+RLAMPL*ZI(J,IRSAVE)
             AMINDL(J)=A(J)+RLAMMN*ZI(J,IRSAVE)
             ABAR(J)=A(J)+RLAMBA*ZI(J,IRSAVE)
 440        CONTINUE
            CALL PCAL(APLUSDL,NMAX,PPLUSDL,JMAX)
            CALL PCAL(AMINDL,NMAX,PMINDL,JMAX)
            CALL PCAL(ABAR,NMAX,P,JMAX)
         CALL EXPGRD(X,Y,P,AAA,CCC,DDD,GGG,WWW,YMIN,
     1                YMAX,IMAX,JMAX,ICOL)
         CALL EXPGRD (X,YPLUSDL,PPLUSDL,AAA,CCC,DDD,GGG,WWW,YMIN,YMAX,
     1                IMAX,JMAX,ICOL)
         CALL EXPGRD (X,YMINDL,PMINDL,AAA,CCC,DDD,GGG,WWW,YMIN,YMAX,
     1                IMAX,JMAX,ICOL)
       CALL BLIMP(X,Y,IMAX,JMAX,U,ITERATE,UE,DELTA,RE)
       CALL BLIMP(X,YPLUSDL,IMAX,JMAX,UPLUSDL,ITERATE,UE,DELTA,RE)
       CALL BLIMP(X,YMINDL,IMAX,JMAX,UMINDL,ITERATE,UE,DELTA,RE)
```

```
      CALL NEWP1(U,P,UPLUSDL,UMINDL,DUUU,RLAMPL,RLAMHN,
     1                RLAMBA,RMSDUUU,H,K,PERCENT,IMAX,JMAX,
     2    ABAR2,NMAX,X,Y,AAA,CCC,DDD,GGG,WWW,YMIN,YMAX,ICOL,
     3   ITERATE,UE,DELTA,P3,Y3,D3U,U3,A,ZI,IRSAVE,WHO,WH2,WH3)
         WRITE(6,900) IRSAVE,(A(J),J=1,NMAX)
  900    FORMAT(1X,'ABOVE RESULTS ARE FOR OUTER OPT AND',
     1          ' IRSAVE=',I5,/,' THE OLD A(I) VALUES ARE=',
     2           5E13.5)
         WRITE (6,901) (ZI(J,IRSAVE),J=1,NMAX)
  901    FORMAT (1X,' THE OLD ZI(J,IRSAVE) VALUES=',5E13.5,/,
     1            1X,' POLD IS CALCULATED BY USING',
     2            ' A=A(OLD)+(RLAMBA OLD)*(ZI OLD)',/,/)
         ERRORL=ABS(RLAMOLD-RLAMBA)*100.0
           IF (RLAMOLD.NE.0.0) ERRORL=ERRORL/ABS(RLAMOLD) GO TO 540
            IF (ERRORL.LT.ERMXL) GO TO 540
  480       CONTINUE
  540    CONTINUE
            DO 490 J=1,NMAX
             A(J)=A(J)+RLAMBA*ZI(J,IRSAVE)
  490       CONTINUE

  550        CONTINUE
  370        CONTINUE
  580        CONTINUE
C
CC
CCC
         CALL POHL(Y,DELTA,UE,IMAX,JMAX,UEXACT)
         WRITE (6,400)
  400    FORMAT (2X,'I',5X,'J',12X,'P',12X,'X',12X,'Y',12X,
     1           'U',12X,'UEXACT',8X,'U-UEXACT')
         DO 610 I=1,IMAX
         IF(I.EQ.2.OR.I.EQ.IMAX) THEN
           DO 613 J=1,JMAX
            WRITE (6,500) I,J,P(J),X(I),Y(I,J),U(I,J),
     1                    UEXACT(I,J),U(I,J)-UEXACT(I,J)
  500    FORMAT (1X,2I5,6E13.5)
  613       CONTINUE
           ENDIF
  610    CONTINUE
         CALL PCAL(A,NMAX,P,JMAX)
         CALL EXPGRD(X,U,P,AAA,CCC,DDD,GGG,WWW,YMIN,
     1               YMAX,IMAX,JMAX,ICOL)
         CALL BLIMP(X,Y,IMAX,JMAX,U,ITERATE,UE,DELTA,RE)
            WRITE (6,600) H,RMSDUUU
  600    FORMAT (1X,'SUM OF THE SQUARES OF DUUU, H=',E13.5,
     1          /,1X,'RMS VALUE OF DUUU, RMSDUUU=',E13.5,/,/)
         CALL ERROR(U,UEXACT,IMAX,JMAX)
         DO 630 I=2,IMAX
          DO 640  J=1,JMAX
           ETA=Y(I,J)*SQRT(RE/X(I))
           WRITE(11,10) ETA,UEXACT(I,J)
  640     CONTINUE
  630    CONTINUE
  10     FORMAT(2E13.5)
         ENDFILE 8
         ENDFILE 9
         ENDFILE 10
         ENDFILE 11
         ENDFILE 12
         ENDFILE 13
         CALL SECOND (CPF)
         CPU=CPF-CPI
         WRITE (6,700) CPU
  700    FORMAT (1X,'CPU=',E13.5)
         STOP
         END
```

```
      SUBROUTINE PCAL (A,NMAX,P,IMAX)
       DIMENSION A(NMAX),P(IMAX)
        DO 10 I=1,IMAX
C         P(I)=A(1)*I+A(2)
          P(I)=A(1)+A(2)
 10      CONTINUE
       RETURN
       END
C
C  THIS SUBROUTINE SOLVES THE UNIFIED DIFF RELATION FOR THE C  GRID EQ
USING A TRIDIAGONAL SOLVER
C
      SUBROUTINE EXPGRD(X,Y,P,A,C,D,G,W,YMIN,YMAX,IMAX,JMAX,L)
       DIMENSION P(JMAX),A(JMAX)
       DIMENSION C(JMAX),D(JMAX),G(JMAX),W(JMAX)
       DIMENSION X(IMAX),Y(IMAX,JMAX)
       WRITE (6,100)
 100   FORMAT (/T15,'GRID CAL',/,T5,'USING A UNIFIED DIFFREL')
       Y(L,1)=YMIN
       Y(L,JMAX)=YMAX
       DO 190 J=2,JMAX-1
        A(J)=EXP(-P(J))/(EXP(-P(J))+1.0)
        C(J)=1/(EXP(-P(J))+1.0)
        D(J)=0.0
 190   CONTINUE
       G(1)=Y(L,1)
       W(1)=0.0
       DO 290 J=2,JMAX-1
         W(J)=-C(J)/(1.0+A(J)*W(J-1))
         G(J)=(D(J)+A(J)*G(J-1))/(1.+A(J)*W(J-1))
 290   CONTINUE
       DO 390 J=JMAX-1,2,-1
        Y(L,J)=G(J)-W(J)*Y(L,J+1)
 390   CONTINUE
      DO 490 I=1,IMAX
       DO 590 J=1,JMAX
        IF(I.EQ.L) THEN
         Y(I,J)=Y(L,J)
        ELSE
         Y(I,J)=Y(L,J)*SQRT(X(I)/X(L))
        ENDIF
 590   CONTINUE
 490  CONTINUE
C      WRITE(14,*) 'ADAPTIVE GRID COORDINATES'
C      WRITE(14,1)((X(I),Y(I,J),I=1,IMAX),J=1,JMAX)
 1     FORMAT(2E13.5)
       WRITE (6,200)
 200   FORMAT(10X,'TRIDIAGONAL SUBROUTINE IS USED FOR GRID EQ')
       RETURN
       END
C
C     THIS SUBROUTINE CALCULATES CONVERSION FOR PHYSCAL TO
C     DORODINITSYN PLANE.  TRAPEZOIDAL INTEGRATION.
C
      SUBROUTINE TRAPZ(H,Y,I,J,IMAX,JMAX,SUM,HEND)
      DIMENSION H(IMAX,0:JMAX+1),Y(IMAX,JMAX)
      SUM=0.0
      DO 10 k=2,J
       F=Y(I,K)-Y(I,K-1)
       AREA= F/2*(H(I,K)+H(I,K-1))/HEND
       SUM=SUM+AREA
10    CONTINUE
      RETURN
      END
```

```
C  THIS SUBROUTINE CALCULATES U VELOCITY FOR THE ITERATION.
C
      SUBROUTINE BLIMP(X,Y,IMAX,JMAX,U,IB,UE,DELTA,RE)
       DIMENSION X(IMAX),Y(IMAX,JMAX),U(IMAX,0:JMAX+1),
     &          V(21,19),H(21,0:60),W(19)
       DIMENSION DX(21),DY(21,19),DYXI(21,19),DYB(21,19)
       DIMENSION ETAX(21,19),RHO(19),RM(19),T(19),DYRE(21,19)
       DIMENSION UNM1(21,19),HNM1(21,19),VNM1(21,19),UIM1(19)
       DIMENSION HIM1(19),UIM2(19),HIM2(19),Q(21),HHREF(21)
       DIMENSION YP(21,19),R(21,19),DELTA(IMAX),DYF(21,19)
       DIMENSION TAU(21),CF(21),UD(21,19),TW(21)
       REAL MU,MUINF,L,M,MINF,MUW,MUEDG,ME,MUSTAR
       DATA NT,ICNT,KT,NTW,IRST/020,010,10,005,0/
       DATA MINF,PINF,TINF/.01,2116.2,530./
C      DATA MINF,PINF,TINF/14.24,.50343,48.6/
C      DATA DT,L,PR,EPS/200.,28.95,1.,.00001/
       DATA CP,RG,GAM/6006.,1715.,1.4/
       DATA FSOR/1.0/
       REWIND 13
       REWIND 12
       REWIND 11
       REWIND 10
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C    THIS PROGRAM SOLVES THE UNSTEADY BOUNDARY LAYER EQUATIONS
C    ALSO IT SOLVES FOR THE HEAT RATE AT THE WALL AND
C    DETERMINES A HEAT TRANSFER COEFFICIENT
C
C      SYMBOLS:
C      AE                EDGE SPEED OF SOUND
C      AINF              INFINITY SPEED OF SOUND
C      CF                SKIN FRICTION COEFFICIENT
C      COSJ              SOR CONSTANT
C      CP                COEFFICIENT OF HEAT(FT-LB/SLUG/R)
C      DEL               CHANGE IN VALUE AFTER ITERATION
C      DELM              MAX DEL
C      DELTA             BOUNDARY LAYER THICKNESS
C      DX                DERIVATIVE OF X WRT XI
C      DY                DERIVATIVE OF Y WRT ETA
C      DYXI              DERIVATIVE OF Y WRT XI
C      DYB               Y DIFFERENCE BETWEEN J AND J-1
C      DYC               CENTRAL Y DIFFERENCE
C      DYF               Y DIFFERENCE BETWEEN J+1 AND J
C      DT                DELTA TIME
C      EPS               CONVERGENCE EPSILON
C      EMAX              MAX ERROR
C      GAM               RATIO OF SPECIFIC HEATS (GAMMA)
C      GGM1              GAMMA/GAMMA-1
C      H                 TOTAL ENTHALPY
C      HC                HEAT COEFFICIENT(BTU/FT2/S/R)
C      HE                EDGE ENTHALPY
C      HNM1              TOTAL ENTHALPY AT OLD TIME LEVEL
C      HIM1              ENTHALPY AT I-1 LOCATION
C      HIM2              ENTHALPY AT I-2 LOCATION
C      HINF              INFINITE ENTHALPY
C      HHREF             H/HREF CALCULATED
C      HREF              REFERENCE HEAT COEFFICIENT(BTU/FT2/S/R)
C      HT                THEORETICAL NONISOTHERMAL HEAT COEFF
C      I                 COUNTER IN XI DIRECTION
C      ICNT              THE NUMBER OF TIMESTEPS BETWEEN PRINTOUTS
C      ICOMP             INCOMPRESSIBLE CASE FLAG
C      IMAX              MAX NUMBER OF STEPS IN XI DIRECTION
C      IMET              NUMERICAL YXI CALCULATION FLAG
C      IPRT              FIELD LISTING FLAG
C      IR                OUTSIDE DELTA CALCULATION FLAG
C      IRST              RESTART INDICATOR
C      J                 COUNTER IN ETA DIRECTION
C      JMAX              MAX NUMBER OF STEPS IN ETA DIRECTION
C      JMM1              JMAX-1
C      K                 COUNTER IN ITERATIONS
C      KT                MAX NUMBER OF ITERATIONS
```

```
C     L               LENGTH OF PLATE (FEET)
C     M               AXISYMMETRIC COEFF.
C     MINF            MACH AT INFINITY
C     MU              VISCOSITY, NONDIMENSIONAL
C     MUEDG           EDGE VISCOSITY
C     MUINF           VISCOSITY AT INFINITY(SLUGS/FT/S)
C     N               COUNTER IN TIME
C     NNTS
C     NT               MAX NUMBER OF TIME STEPS
C     NTS
C     NTW              NUMBER OF TIME STEPS TO CALCULATE OMEGA
C     PHI             NONISOTHERMAL CORRECTION FACTOR
C     PINF            PRESSURE INFINITY
C     PRESS           NONDIMENSIONALIZED EDGE PRESSURE
C     PRESSO          NONDIMENSIONALIZED INITIAL PRESS
C     PR              PRANDTL NUMBER
C     PE              PRESSURE AT THE EDGE OF B.L.(LB/FT2)
C     Q               HEAT RATE AT THE WALL(BTU/FT2/S)
C     RE              REYNOLD'S NUMBER ON L
C     REE             EDGE REYNOLD@S NUMBER ON L
C     RG              GAS CONSTANT (FT-LB/SLUG-R)
C     RHO             DENSITY
C     RHOE            EDGE DENSITY
C     RM              DENSITY TIMES VISCOSITY
C     RE              REYNOLD'S NUMBER
C     RINF            DENSITY AT INFINITY (SLUG/FT**3)
C     T               TEMPERATURE
C     TAU             SKIN FRICTION(LB/FT2)
C     TEQ             EQUIVALENT TEMPERATURE(R)
C     THETA           WEDGE/CONE HALF ANGLE
C     TIME            TOTAL TIME ACCUMULATED
C     TE              TEMPERATURE AT THE EDGE OF B.L.(R)
C     TINF            TEMPERATURE AT INFINITY (DEG RANKINE)
C     TOINF           TOTAL TEMPERATURE AT INFINITY (RANKINE)
C     TS
C     TW              TEMPERATURE AT THE WALL(R)
C     TW1             TEMPERATURE AT WALL BEFORE PANEL
C     TW2             TEMPERATURE AT WALL AT PANEL
C     U               VELOCITY IN X DIRECTION
C     UNM1            VELOCITY, X DIRECTION AT OLD TIME LEVEL
C     UE              VELOCITY AT THE EDGE OF B.L.(FT/S)
C     UIM1            VELOCITY VECTOR AT I-1 LOCATION
C     UIM2            VELOCITY VECTOR AT I-2 LOCATION
C     UINF            VELOCITY AT INFINITY CONDITION (FT/SEC)
C     V               VELOCITY IN Y DIRECTION
C     VNM1            VELOCITY,Y DIRECTION AT OLD TIME LEVEL
C     W               OPTIMIZATION OMEGA
C     XTW2            POSITION OF PANEL ON WEDGE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     WRITE(6,*)'NUMERICAL YXI METRIC AFTER IMET='
      READ(5,*)IMET
C     WRITE(6,*)'L(FT)=, PR='
      READ(5,*)L,PR
C     WRITE(6,*) 'INPUT: CONE=1.,WEDGE=0.'
      READ(5,*) M
C     WRITE(6,*) 'IR=0,R=F(X) ONLY'
C     WRITE(6,*) 'IR='
      READ(5,*) IR
C     WRITE(6,*)'MINF,PINF,TINF='
      READ(5,*)MINF,PINF,TINF
C     WRITE(6,*)'TW1,TW2,XTW2='
      READ(5,*)TW1,TW2,XTW2
C     WRITE(6,*)'IPRINT=0 NO LISTING ON FILE FIELD'
C     WRITE(6,*)'IPRT='
      READ(5,*)IPRT
C     WRITE(6,*)'NT,ICNT,KT,IRST,NTW='
      READ(5,*)NT,ICNT,KT,IRST,NTW
C     WRITE(6,*)'DT,EPS='
      READ(5,*)DT,EPS
```

```
C      WRITE(6,*) 'THETA,BETA GUESS?'
       READ(5,*) THETA,PO
       PI=ACOS(-1.)
       JMM1=JMAX-1.
       GGM1=GAM/(GAM-1.)
       MUINF=(TINF**1.5*2.2685E-8)/(TINF+198.6)
       AINF=SQRT(GAM*RG*TINF)
       UINF=MINF*AINF
       RINF=PINF/(RG*TINF)
       TOINF=TINF*(1+(GAM-1.)*.5*MINF**2)
       HINF=CP*TINF+UINF**2/2
       COSJ=COS(PI/FLOAT(JMAX))*FSOR
       RE=RINF*UINF*L/MUINF
       THETA=THETA*PI/180.
       IF(THETA.NE.0.0.AND.MINF.GE.1.0)THEN
        CALL BETA(THETA,MINF,PINF,TINF,PO,CP,RG,ME,PE,TE,
      &           HE,UE)
        AE=SQRT(GAM*RG*TE)
        MUEDG=(TE**1.5*2.2685E-8)/(TE+198.6)
        RHOE=PE/(RG*TE)
        REE=RHOE*UE*L/MUEDG
       ELSE
        AE=AINF
        ME=MINF
        PE=PINF
        TE=TINF
        HE=HINF
        UE=UINF
        RHOE=RINF
        REE=RE
        MUEDG=MUINF
       ENDIF
       HEND=HE/UINF**2
       IF(IB.EQ.0) THEN
        WRITE(9,*)'RE=,PR= ',RE,PR
        WRITE(9,*)'RINF,UINF,MUINF=',RINF,UINF,MUINF
        WRITE(9,*)'MINF,AINF,TINF,PINF=',MINF,AINF,TINF,PINF
        WRITE(9,*)'ME,PE,TE,AE,UE=',ME,PE,TE,AE,UE
        WRITE(9,*) 'NT,ICNT,KT,IRST,DT,EPS,NTW,IMET=,',
      1             NT,ICNT,KT,IRST,DT,EPS,NTW,IMET
       ENDIF
       PRESS=PE/(RINF*UINF**2)
       PRESSO=PRESS
       HREF=.332*CP*SQRT(MUINF*RINF*UINF)/PR**(2./3.)
C
C      GENERATE A GRID AND INITIAL CONDITIONS
C
       CALL FIRST(IMAX,JMAX,REE,CP,TW,TE,UE,UINF,X,Y,U,V,H,PR,
      &           XTW2,TW1,TW2,IMET,PE,RG,M,RHOE,MINF,DELTA,L)
       IF (Y(IMAX,JMAX).LT.DELTA(IMAX).AND.MINF.GT..05)
      1    GO TO 920
       DO 15 I=1,IMAX
         U(I,0)=0.
         H(I,0)=0.0
         U(I,JMAX+1)=UE/UINF
15       H(I,JMAX+1)=HEND
C
        DO 20 I=1,IMAX
         U(I,JMAX)=UE/UINF
20       H(I,JMAX)=HEND
C
C      CALCULATE METRICS
       CALL METRIC(X,Y,RE,IMET,DX,DY,DYXI,DYB,DYC,DYF,ETAX,
      &            DYRE,IMAX,JMAX,JMM1)
C
C      PUT INFORMATION INTO THE OLD TIME LEVEL INITIALLY
C
       DO 30 I=1,IMAX
       DO 30 J=1,JMAX
          UNM1(I,J)=U(I,J)
```

```
          VNM1(I,J)=V(I,J)
30        HNM1(I,J)=H(I,J)
        DO 40 J=1,JMAX
         IF (J.LE.11) THEN
          W(J)=1.6-(J-1)*.06
         ELSE
          W(J)=1.0
         ENDIF
40      CONTINUE
C
C       START THE TIME LOOP
        TIME=0.0
        DO 100 N=1,NT
         TIME=TIME+DT
         DPDT=(PRESS-PRESSO)
         PRESSO=PRESS
C
C       START BACK AT BEGINNING OF GRID--REINITIALIZE VECTORS
C
120      DO 130 J=1,JMAX
          UIM1(J)=0.0
130       HIM1(J)=0.0
C
CALCULATE THE OPTIMIZATION FACTOR OMEGA EVERY NTW TIME STEPS
C
         IF(N.EQ.(N/NTW)*NTW)THEN
         CALL MURHO(H,U,GGM1,PRESS,UINF,CP,MUINF,IMAX,JMAX,RM,
     1              W,TE)
         CALL OMEGA(U,V,DT,DX,DY,DYB,DYF,DYRE,ETAX,RM,W,WN,
     &             IMAX,JMM1,JMAX,COSJ)
         ENDIF
         IF(IR.EQ.1) CALL RADIUS(X,Y,DELTA,IMAX,JMAX,THETA,R,IR)
C
C       START I LOOP--MARCH IN XI DIRECTION
C
        DO 200 I=2,IMAX
         IM1=I-1
         IM2=I-2
         DELM=0.0
C       PUT NEW INFORMATION INTO THE VELOCITY AND ENTHALPY
C       VECTORS AT THE I-1 AND I-2 LOCATIONS
C
          DO 210 J=1,JMAX
           UIM2(J)=UIM1(J)
           UIM1(J)=U(I-1,J)
           HIM2(J)=HIM1(J)
210        HIM1(J)=H(I-1,J)
C
C       DEFINE CONSTANTS TO DETERMINE THE COEFFICIENTS FOR THE
C       U AND V VELOCITY EQUATIONS AND THE ENTHALPY EQUATION
C
          IF(I.EQ.2)THEN
            CIJ1=1.
            CIM1=1.
            CIM2=0.
            CVIJ=-.5
            CVIM1=.5
            CVIM2=0.
          ELSE
            CIJ1=1.5
            CIM1=2.0
            CIM2=-.5
            CVIJ=-.75
            CVIM1=1.
            CVIM2=-.25
          ENDIF
C
C       START THE ITERATION LOOP
C
        DO 300 K=1,KT
```

```
        EMAX=0.0
        EUMAX=0.
        EVMAX=0.
        EHMAX=0.
C
C     CALCULATE NEW VALUES FOR TEMPERATURE,DENSITY, AND VISCOSITY USING
C     THE NEW VALUES OF U, V, AND H (CALCATED AT THE OLD ITERATION LEVEL)
C
       DO 310 J=1,JMAX
          T(J)=H(I,J)-U(I,J)**2/2.
          TEMP=T(J)*UINF**2./CP
          RHO(J)=GGM1*PRESS/T(J)
      IF(TE.LT.200.) THEN
       MU=TEMP**.5*2.32E-8/(1.+220./(TEMP*10.**(9./TEMP)))/MUINF
      ELSE
       MU=(TEMP**1.5*2.2685E-8)/(TEMP+198.6)/MUINF
      ENDIF
310     RM(J)=RHO(J)*MU
C
C     START J LOOP--MARCHING IN ETA DIRECTION
C
      DO 400 J=JMM1,2,-1
        JM1=J-1
        JM2=J-2
        JP=J+1
        UOLD=U(I,J)
        HOLD=H(I,J)
        IF(K.EQ.1)THEN
          WOPT=1.0
        ELSE
          WOPT=W(J)
        ENDIF
C
C     SETUP CONSTANTS TO DETERMINE COEFFICIENTS IN U, V, AND H EQUATIONS
C
C     SOLVE FOR U,V,AND H
C
        DRMF=(RM(JP)+RM(J))/DYF(I,J)*.5
        DRMB=(RM(J)+RM(JM1))/DYB(I,J)*.5
        CON=ETAX(I,J)*U(I,J) + V(I,J)/DY(I,J)
        IF(CON.LT.0.0)THEN
          IF(J.EQ.JMM1) THEN
          CIJ2=-1.0
          CJM1=-1.0
          CJM2=0.0
          ELSE
          CIJ2=-1.5
          CJM1=-2.0
          CJM2=.5
          ENDIF
          DRM1=DRMF
          DRM2=DRMB
          J1=JP
          J2=J+2
          J3=JM1
        ELSE
        IF(J.EQ.2)THEN
          CIJ2=1.
          CJM1=1.
          CJM2=0.
        ELSE
          CIJ2=1.5
          CJM1=2.0
          CJM2=-.5
        ENDIF
        DRM1=DRMB
        DRM2=DRMF
        J1=JM1
        J2=JM2
        J3=JP
```

```
          ENDIF
C         SOLVE FOR U VELOCITY
          CIJ=1. + CIJ1*DT/DX(I)*U(I,J) + CIJ2*DT*CON
     &         + DT*DYRE(I,J)*(DRMF+DRMB)
          CONJ1=CJM1*DT*CON + DT*DYRE(I,J)*DRM1
          CONJ2=CJM2*DT*CON
          CONIM=DT/DX(I)*U(I,J)
          CONVS=DT*DYRE(I,J)*DRM2
          US=(UNM1(I,J) + CIM1*CONIM*UIM1(J) + CIM2*CONIM*UIM2(J) +
     &       CONJ1*U(I,J1) + CONJ2*U(I,J2) + CONVS*U(I,J3))/CIJ
          U(I,J)=WOPT*US + (1.-WOPT)*U(I,J)
C         SOLVE FOR ENTHALPY
          CON=ETAX(I,J)*U(I,J) + V(I,J)/DY(I,J)
          IF(CON.LT.0.0)THEN
            IF(J.EQ.JMM1) THEN
              CIJ2=-1.0
              CJM1=-1.0
              CJM2=0.0
              ELSE
              CIJ2=-1.5
              CJM1=-2.0
              CJM2=.5
              ENDIF
              DRM1=DRMF
              DRM2=DRMB
              J1=JP
              J2=J+2
              J3=JM1
            ELSE
            IF(J.EQ.2)THEN
              CIJ2=1.
              CJM1=1.
              CJM2=0.
            ELSE
              CIJ2=1.5
              CJM1=2.0
              CJM2=-.5
            ENDIF
            DRM1=DRMB
            DRM2=DRMF
            J1=JM1
            J2=JM2
            J3=JP
            ENDIF
            CIJ=1. + CIJ1*DT/DX(I)*U(I,J) + CIJ2*DT*CON
     &         + DT*DYRE(I,J)*(DRMF+DRMB)/PR
            CONJ1=CJM1*DT*CON + DT*DYRE(I,J)*DRM1/PR
            CONJ2=CJM2*DT*CON
            CONIM=DT/DX(I)*U(I,J)
            CONVS=DT*DYRE(I,J)*DRM2/PR
            TCON=DYRE(I,J)*DT*((PR-1.)/PR)*.5
            HS=(HNM1(I,J) + CIM1*CONIM*HIM1(J) + CIM2*CONIM*HIM2(J) +
     &         CONJ1*H(I,J1) + CONJ2*H(I,J2) + CONVS*H(I,J3) +
     &         DPDT/RHO(J)+TCON*(DRMF*U(I,JP)**2-(DRMF+DRMB)*U(I,J)**2+

     &         DRMB*U(I,JM1)**2))/CIJ
            H(I,J)=WOPT*HS + (1.-WOPT)*H(I,J)
C
C         COMPUTE THE ERROR
C
            EU=ABS(U(I,J)-UOLD)
            EH=ABS(H(I,J)-HOLD)
            IF(EU.GE.EUMAX)EUMAX=EU
            IF(EH.GE.EHMAX)EHMAX=EH
          IF(EUMAX.GE.EMAX)EMAX=EUMAX
          IF(EHMAX.GE.EMAX)EMAX=EHMAX
C
C         END J LOOP
400       CONTINUE
C         CALCULATE V VELOCITY
```

```
      DO 320 J=2,JMM1
        JM1=J-1
        VOLD=V(I,J)
C     IF (IR.EQ.0) THEN
        V(I,J)=V(I,JM1)+(.5*(DYXI(I,J)+DYXI(I,JM1))*
     1          (U(I,J)-U(I,JM1))+(Y(I,J)-Y(I,JM1))*
     2          (CVIJ*(U(I,J)+U(I,JM1))+
     3          CVIM1*(UIM1(J)+UIM1(JM1))+
     4          CVIM2*(UIM2(J)+UIM2(JM1))))/DX(I)
C     ELSE
C     V(I,J)=(1/R(I,J)**M)*(R(I,JM1)**M*V(I,JM1) + R(I,J)**M
C    &*VCNIJ*U(I,J)+R(IM1,J)**M*CVIM1*D1*UIM1(J) + R(IM2,J)**M
C    &*CVIM2*D1*UIM2(J)+R(I,JM1)**M*VCNJ1*U(I,JM1)+R(IM1,JM1)**M
C    &*CVIM1*D1*UIM1(JM1)+R(IM2,JM1)**M*CVIM2*D1*UIM2(JM1))
C     ENDIF
      EV=ABS(V(I,J)-VOLD)
      IF(EV.GE.EVMAX)EVMAX=EV
      IF(EVMAX.GE.EMAX)EMAX=EVMAX
320   CONTINUE
C
C     CHECK CONVERGENCE
C
      IF(EMAX.LE.EPS) GO TO 420
C
C     END ITERATION LOOP
300   CONTINUE
420   DO 430 J=2,JMM1
       DEL=ABS(U(I,J) - UNM1(I,J))
       IF(DEL.GT.DELM)DELM=DEL
       DEL=ABS(H(I,J) - HNM1(I,J))
       IF(DEL.GT.DELM)DELM=DEL
430   CONTINUE
      V(I,JMAX)=V(I,JMM1)
      DO 330 J=1,JMAX
       U(1,J)=U(2,J)
       H(1,J)=H(2,J)
330   V(1,J)=V(2,J)
C     MOVE THE RESULTS INTO THE OLD TIME LEVEL
C
      DO 440 J=1,JMAX
       UNM1(I,J)=U(I,J)
       HNM1(I,J)=H(I,J)
440   VNM1(I,J)=V(I,J)
C
C     CALCULATE THE HEAT RATE AT THE WALL
C
      FDELY=(-3.*Y(I,1) + 4.*Y(I,2) - Y(I,3))/2.
      IF (TE.LT.200.) THEN
      MUW=2.32E-8*TW(I)**.5/(1.+220./(TW(I)*10.**(9./TW(I))))
      ELSE
      MUW=(TW(I)**1.5*2.2685E-8)/(TW(I)+198.6)
      ENDIF
C     CONDUCTIVITY IS EQUAL TO MUW*CP/PR=K
      Q(I)=MUW*RHO(1)*(-3.*T(1) + 4.*T(2) - T(3))* RINF
     &          *UINF**2/(2.*L*PR*FDELY*RHOE)
C
C TAW SHOULD USUALLY BE BASED ON EDGE CONDITIONS FORMALLY
C BUT CAN BE BASED ON FREESTREAM IF YOU ARE CONSISTENT!!!!
C
      TAW=TE*(1.+SQRT(PR)*(GAM-1.)*.5*ME**2)
      HC=Q(I)/(TAW - TW(I))
      HHREF(I)=HC/HREF*SQRT(X(I)*L)
      TAU(I)=MUW*UINF*(-3*U(I,1)+4*U(I,2)-U(I,3))*.5
     1    /(L*FDELY*RHO(1))
      CF(I)=2.*TAU(I)/(RINF*UINF**2)
C
C     END I LOOP
200   CONTINUE
C
C     WRITE OUT SOLUTION AT A TIME STEP
```

```
C
      IF(N.EQ.1.OR.N.EQ.(N/ICNT)*ICNT)THEN
      WRITE(10,1)N,TIME
      WRITE(6,*)'DELT=',DELM,'K=',K,'U,V,H ERR=',EUMAX,EVMAX,EHMAX
      WRITE(10,*)'DELT=',DELM,'K=',K,'U,V,H ERR=',EUMAX,EVMAX,EHMAX
      WRITE(9,12)
      DO 150 I=1,IMAX
      IF (I.EQ.2.OR.I.EQ.IMAX) THEN
      WRITE(10,2)I,X(I)
      IF(IPRT.EQ.0)GO TO 161
      WRITE(10,3)
      DO 160 J=1,JMAX
      T(J)=H(I,J)-U(I,J)**2/2.
      TEMP=T(J)*UINF**2/CP
      WRITE(10,4)J,Y(I,J),U(I,J),H(I,J),T(J),TEMP,V(I,J),RHO(J)
160   CONTINUE
      ENDIF
161   IF(I.EQ.1) GO TO 150
      WRITE(10,5) Q(I),HHREF(I)
      TIMEP=TIME*L/UINF
C CF RATIO TO BLASIUS CAN BE BASED ON EDGE OR FREESTREAM RE
      CFCF=CF(I)/.664*SQRT(REE*X(I))
      WRITE(9,7)TIMEP,X(I),CFCF,HHREF(I),TW(I)
      IF(MINF.LE..05.AND.N.EQ.NT) THEN
      WRITE(13,10) X(I),CFCF
      WRITE(12,10) X(I),HHREF(I)
      ENDIF
150   CONTINUE
      ENDIF
C     END TIME LOOP
100   CONTINUE
C
C     COMPUTE PHYSICAL Y VALUES
      IF (MINF.LE..05) THEN
      DO 700 I=2,IMAX
      DO 710 J=1,JMAX
      ETA=Y(I,J)*SQRT(RE/X(I))
      WRITE(11,10) ETA,U(I,J)
 710  CONTINUE
 700  CONTINUE
      GO TO 920
      ENDIF
      DO 800 J=1,JMAX
      YP(1,J)=0.0
C     WRITE(11,10)X(1),YP(1,J)
800   CONTINUE
C     WRITE(13,11)
      DO 810 I=2,IMAX
       DO 820 J=1,JMAX
       T(J)=H(I,J) - U(I,J)**2/2
820   RHO(J)=GGM1*PRESS/T(J)
C
      YP(I,1)=0.0
C     WRITE(11,10)X(I),YP(I,1)
       DO 830 J=2,JMAX
C       YP(I,J)=YP(I,J-1) + (1/RHO(J) + 1/RHO(J-1))/2*(Y(I,J)-Y(I,J-1))
        CALL TRAPZ(H,Y,I,J,IMAX,JMAX,SUM,HEND)
        YP(I,J)=SUM
        ETAC=YP(I,J)*SQRT(UE*RHOE*L/(MUEDG*X(I)))
C       WRITE(11,10)X(I),YP(I,J)
        T(J)=T(J)*UINF**2./(CP*TE)
        UD(I,J)=U(I,J)*UINF/UE
        WRITE(11,10) ETAC,T(J)
        WRITE(13,10) ETAC,UD(I,J)
C       WRITE(13,8) T(J),U(I,J),ETAC
830    CONTINUE
810   CONTINUE
      DO 825 I=1,IMAX
       DO 826 J=1,JMAX
        WRITE(11,10) X(I),Y(I,J)
```

```
 826   CONTINUE
 825  CONTINUE
C
C     THIS SECTION FIGURES OUT THE THEORETICAL NONISOTHERMAL VALUES
C     FOR H/HREF
C
C CAUTION! TSTAR BASED ON TW1! DO YOU WANT CONSTANT REFERENCE
C          OR ACTUAL TW?
840   TAW=TE*(1.+SQRT(PR)*(GAM-1.)*.5*ME**2)
      TSTAR=TE+.5*(TW1-TE)+.22*(TAW-TE)
      RSTAR=PE/(RG*TSTAR)
      IF(TE.LT.200.) THEN
       MUSTAR=(2.32E-8*TSTAR**.5/(1.+220./(TSTAR*10.**(9./TSTAR))))
      ELSE
       MUSTAR=(TSTAR**1.5*2.2685E-8)/(TSTAR+198.6)
      ENDIF
      DO 900 I=1,IMAX
       HISO=.332*CP*SQRT(MUSTAR*RSTAR*UE)*SQRT(X(I)*L)/(PR**(2./3.)*HREF)
       WRITE(9,*) X(I),'HISO=',HISO
 900  CONTINUE
 920  CONTINUE
1     FORMAT(///,2X,'N=',I6,5X,'TIME=',F15.6,1X,'SEC')
2     FORMAT(//,2X,'I=',I3,5X,'X-LOCATION=',F13.5)
3     FORMAT(/,3X,'J',7X,'Y-LOC',10X,'U',11X,'H',11X,'T',11X,
     &'TEMP(R)',11X,'V',11X,'RHO')
4     FORMAT(1X,I3,5F13.5,2F15.8)
5     FORMAT(2X,'HEAT RATE (Q)=',F13.6,5X,'H/HREF=',F13.6)
6     FORMAT('THIS SOLUTION CONVERGED WITH KT ITERATIONS')
7     FORMAT(2X,F15.5,4(2X,F13.5))
8     FORMAT(3(2X,E13.5))
9     FORMAT(34X,2(2X,F13.5))
10    FORMAT(2E13.5)
11    FORMAT(5X,'T/TINF',9X,'U/UINF',11X,'ETAC')
12    FORMAT(/,7X,'TIME',15X,'X',13X,'CF/CF',11X,'HHREF',
     1       10X,'TWALL')
      REWIND 5
      RETURN
      END
C
C     THIS SUBROUTINE CALCULATES BETA AND EDGE CONDITIONS
C
      SUBROUTINE BETA(THETA,MINF,PINF,TINF,PO,CP,RG,ME,PE,TE,
     &HE,UE)
C     CALCULATES EDGE CONDITIONS GIVEN DEFLECTION ANGLE
C     SHOCK ANGLE CALCULATED TO 1.0E-08 TOLERANCE
      REAL MINF,ME,HE
      Q(X)=((GAM+1.)/2.*SIN(X)*SIN(THETA)/COS(X-THETA)+
     &(1./MINF**2.))**.5-SIN(X)+X
      PI=ACOS(-1.)
      GAM=1.4
      TOL=1.0E-08
      PO=PO*PI/180.
      Y=Q(PO)-PO
      DO 10 I=1,50
       XNEW=Q(PO)
       Y=Q(XNEW)-XNEW
       EROR=ABS(XNEW-PO)/ABS(XNEW)
       IF(EROR.LT.TOL.OR.ABS(Y).LT.TOL) GO TO 30
       PO=XNEW
 10   CONTINUE
 30   ME=((2.+(GAM-1.)*(MINF*SIN(XNEW))**2.)/((2.*GAM*(MINF*
     &SIN(XNEW))**2.-(GAM-1.))*(SIN(XNEW-THETA))**2.))**.5
      TE=TINF*(2.+(GAM-1.)*(MINF*SIN(XNEW))**2.)/(2.
     & +(GAM-1.)*(ME*SIN(XNEW-THETA))**2.)
      UE=ME*SQRT(GAM*RG*TE)
      HE=CP*TE+UE**2/2
      PE=(1+(2*GAM/(GAM+1))*((MINF*SIN(XNEW))**2-1))*PINF
      RETURN
      END
```

```
C     THIS SUBROUTINE TAKES THE BLASIUS SOLUTION, MODIFIES IT BY
C     THE DENSITY, AND FORMS A GRID.  IT ALSO DETERMINES INITIAL
C     CONDITIONS
      SUBROUTINE FIRST(IMAX,JMAX,RE,CP,TW,TE,UE,UINF,
     &  X,Y,U,V,H,PR,XTW2,TW1,TW2,IMET,PE,RG,M,RHOE,MINF,DELTA,L)
      DIMENSION X(IMAX),Y(IMAX,JMAX),U(IMAX,0:JMAX+1),H(IMAX,0:JMAX+1)
      DIMENSION V(IMAX,JMAX),TW(IMAX),DELTA(IMAX)
      REAL M,MINF,MUW,L
C
C  TO USE ANALTICAL METRIC FOR YXI FOR I=1 TO IMET
C  Y POINTS AT I=IMET ARE SCALED BY SQRT(X)
C  THIS WILL NOT GUARANTEE A SMOOTH GRID PAST I=IMET
      DO 30 I=1,IMET
      DO 30 J=1,JMAX
      Y(I,J)=Y(IMET,J)*SQRT(X(I)/X(IMET))
30    CONTINUE
C
C     DEFINE THE WALL TEMPERATURES-- IF TW1=TW2 NO TEMP STEP
C                                    IF TW1<TW2 TEMP STEP
C
      DO 40 I=1,IMAX
       IF(X(I).LE.XTW2)THEN
        TW(I)=TW1
       ELSE
        TW(I)=TW2
       ENDIF
40    CONTINUE
      HE=(CP*TE+UE**2./2.)/UINF**2.
      DELTA(1)=0.0
      DO 50 I=2,IMAX
       HW=CP*TW(I)/UINF**2
       HSUB=HE - HW
       IF (MINF.LE..05)THEN
        DELTA(I)=5.2/SQRT(RE)*SQRT(X(I))
        DELTAT=DELTA(I)/(1.026*PR**(1./3.))
       ELSE
        RHOW=PE/(RG*TW(I))
        IF(TE.LT.200.) THEN
         MUW=2.32E-8*TW(I)**.5/(1.+220./(TW(I)*10.**(9./TW(I))))
         ELSE
         MUW=(TW(I)**1.5*2.2685E-8)/(TW(I)+198.6)
        ENDIF
       DELTA(I)=(280*RHOW*MUW*X(I)/((13.*RHOE*RHOE*L*UE*(1+2*M))))**.5
       DELTAT=DELTA(I)
      ENDIF
      DO 60 J=1,JMAX
       IF(Y(I,J).LE.DELTA(I))THEN
        U(I,J)=UE/UINF*(1.5*Y(I,J)/DELTA(I) - .5*(Y(I,J)/DELTA(I))**3.)
       ELSE
        U(I,J)=UE/UINF
       ENDIF
        V(I,J)=0.0
       IF(Y(I,J).LE.DELTAT)THEN
        H(I,J)=HW + HSUB*(1.5*Y(I,J)/DELTAT - .5*(Y(I,J)/DELTAT)**3)
     & + (U(I,J)/UINF)**2./2. + (UE/UINF)**2./2*
     & (1.5*Y(I,J)/DELTAT-.5*(Y(I,J)/DELTAT)**3.)
       ELSE
       H(I,J)=HE
       ENDIF
60    CONTINUE
50    CONTINUE
      DO 70 J=1,JMAX
       U(1,J)=U(2,J)
       V(1,J)=0.0
       H(1,J)=H(2,J)
70    CONTINUE
      RETURN
      END
```

```
C     COMPUTE THE DENSITY TIMES VISCOSITY FOR EVERY J LOCATION
C
      SUBROUTINE MURHO(H,U,GGM1,PRESS,UINF,CP,MUINF,IMAX,JMAX,
     &RM,W,TE)
      DIMENSION H(IMAX,0:JMAX+1),U(IMAX,0:JMAX+1),RM(JMAX),W(JMAX)
      REAL MU,MUINF
      DO 10 I=2,IMAX
       DO 20 J=1,JMAX
        TEMP=H(I,J)-U(I,J)**2/2.
        DEN=GGM1*PRESS/TEMP
        TEMP=TEMP*UINF**2/CP
        IF(TE.LT.200.)THEN
         MU=(2.32E-8*TEMP**.5/(1.+220./(TEMP*10.**(9./TEMP))))/MUINF
        ELSE
         MU=(TEMP**1.5*2.2685E-8)/(TEMP+198.6)/MUINF
        ENDIF
        RM(J)=DEN*MU
20     W(J)=1.85
10    CONTINUE
      RETURN
      END
C
C
C     THIS SUBROUTINE CALCULATES THE SOR OMEGA
C
      SUBROUTINE OMEGA(U,V,DT,DX,DY,DYB,DYF,DYRE,ETAX,RM,
     &W,WN,IMAX,JMM1,JMAX,COSJ)
      DIMENSION U(IMAX,0:JMAX+1),V(IMAX,JMAX),DX(IMAX),DY(IMAX,JMAX)
      DIMENSION DYB(IMAX,JMAX),DYF(IMAX,JMAX),DYRE(IMAX,JMAX)
      DIMENSION RM(JMAX),W(JMAX),ETAX(IMAX,JMAX)
      REAL LAM
      DO 10 I=2,IMAX
       DO 20 J=2,JMM1
        JP=J+1
        JM=J-1
        DIV=1./DT + 1.5*(U(I,J)/DX(I) + ABS(U(I,J)*ETAX(I,J) +
     &    V(I,J)/DY(I,J))) + DYRE(I,J)*.5*
     &    ((RM(JP)+RM(J))/DYF(I,J) + ((RM(J)+RM(JM))/DYB(I,J)))
       LAM=(-2.*SQRT(ABS(ABS((ETAX(I,J)*U(I,J) + V(I,J)/DY(I,J)) +
     &    .5*DYRE(I,J)*((RM(J)+RM(JM))/DYB(I,J))) *
     &    (.5*DYRE(I,J)*((RM(JP)+RM(J))/DYF(I,J))))))/DIV*COSJ
       IF(LAM.GT.0.97)LAM=0.97
       WN=2./(1.+SQRT(1.-LAM**2))
       IF(WN.LT.W(J))W(J)=WN
20    CONTINUE
10    CONTINUE
      RETURN
      END
C
C     COMPUTES H3 FOR A CONE-ASSUME CONSTANT TRANSVERSE RADIUS
C     ASSUME YCOS(ALPHA) SMALL. IF NOT DELETE NEXT LINE.
C
      SUBROUTINE RADIUS(X,Y,DELTA,IMAX,JMAX,THETA,R,IR)
      DIMENSION X(IMAX),Y(IMAX,JMAX),R(IMAX,JMAX),DELTA(IMAX)
      DO 10 I=1,IMAX
       DO 20 J=1,JMAX
        IF(Y(I,J).LE.DELTA(I)) THEN
         R(I,J)= X(I)*SIN(THETA)
        ELSE
         IF(IR.EQ.0) THEN
          R(I,J)=X(I)*SIN(THETA)
         ELSE
          R(I,J)=X(I)*SIN(THETA)+Y(I,J)*COS(THETA)
         ENDIF
        ENDIF
       IF(R(I,J).EQ.0.0)  R(I,J)=1.E-8
20     CONTINUE
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE METRIC(X,Y,RE,IMET,DX,DY,DYXI,DYB,DYC,DYF,
     &                  ETAX,DYRE,IMAX,JMAX,JMM1)
C     CALCULATES DERIVATIVES XI WRT X, ETA WRT Y, AND ETA WRT X
C     XI WRT X = 1/DX,ETA WRT Y=1/DY,AND ETA WRT X= -DYXI/(DX*DY)
      DIMENSION X(IMAX),Y(IMAX,JMAX),DX(IMAX),DY(IMAX,JMAX)
      DIMENSION DYXI(IMAX,JMAX),DYB(IMAX,JMAX)
      DIMENSION DYF(IMAX,JMAX),ETAX(IMAX,JMAX),DYRE(IMAX,JMAX)
C
      DX(1)=.5*(-3.*X(1)+4.*X(2)-X(3))
      DO 10 I=2,IMAX
       IM1=I-1
       IM2=I-2
       IP=I+1
       IF(I.EQ.IMAX)THEN
        DX(I)=.5*(3.*X(I)-4.*X(IM1)+X(IM2))
       ELSE
        DX(I)=(X(IP)-X(IM1))/2.
       ENDIF
       DYXI(I,1)=0.0
       IF (I.GT.IMET) THEN
C       DYXI(I,JMAX)=Y(I,JMAX)-Y(IM1,JMAX)
        DYXI(I,JMAX)=.5*(3.*Y(I,JMAX)-4.*Y(IM1,JMAX)+Y(IM2,JMAX))
       ELSE
        DYXI(I,JMAX)=Y(I,JMAX)*.5*DX(I)/X(I)
       ENDIF
      DO 10 J=2,JMM1
       JP=J+1
       JM1=J-1
       JM2=J-2
       DY(I,J)=(Y(I,JP)-Y(I,JM1))/2
       IF(I.GT.IMET)THEN
C       DYXI(I,J)=Y(I,J)-Y(IM1,J)
        DYXI(I,J)=.5*(3.*Y(I,J)-4.*Y(IM1,J)+Y(IM2,J))
       ELSE
        DYXI(I,J)=Y(I,J)*.5*DX(I)/X(I)
       ENDIF
       DYB(I,J)=Y(I,J)-Y(I,JM1)
       DYC=(Y(I,JP)-Y(I,JM1))/2.
       DYF(I,J)=Y(I,JP)-Y(I,J)
       ETAX(I,J)=-DYXI(I,J)/(DX(I)*DY(I,J))
10    DYRE(I,J)=1./(DYC*RE)
      RETURN
      END
C
C     THIS SUBROUTINE CALCULATES THE ERROR IN U-UEXACT
C
      SUBROUTINE ERROR(U,UEXACT,IMAX,JMAX)
      DIMENSION U(IMAX,0:JMAX+1),UEXACT(IMAX,0:JMAX+1)
      URMS=0.0
      DO 10 I=1,IMAX
       DO 20 J=1,JMAX
        UER=U(I,J)-UEXACT(I,J)
        URMS=URMS+UER**2.
20     CONTINUE
10    CONTINUE
      RMSU=SQRT(URMS/(IMAX*JMAX))
      WRITE(6,*) 'RMS VALUE OF U-VELOCITY ERROR =',RMSU
      RETURN
      END
```

```
      SUBROUTINE NWP(U,DUUU,H,IMAX,JMAX,ICOL,WHO,WH2,WH3)
       DIMENSION U(IMAX,0:JMAX-1) DUUU(JMAX)
       RMSDU= 0.0
       RMSDUU=0.0
       RMSDUUU=0.0
       HO=0.0
       H2=0.0
       H3=0.0
       WRITE (6,100)
  100  FORMAT (3X,'ETA',07X,'DU',9X,'DUU',10X,'DUUU',7X,'DUUU**2')
         DO 10 I=1,IMAX
C      I=ICOL
       IF(I.EQ.2.OR.I.EQ.IMAX) WRITE(6,*)'I=',I
        DO 20 J=1,JMAX
         IF (J.EQ.1.OR.J.EQ.JMAX) THEN
          IF (J.EQ.1) THEN
           DU=.5*(-3.*U(I,1)+4.*U(I,2)-U(I,3))
           DUU=U(I,1)-2.*U(I,2)+U(I,3)
          ELSE
           DU=.5*(3.*U(I,JMAX)-4.*U(J,JMAX-1)+U(I,JMAX-2))
           DUU=U(I,JMAX)-2.*U(I,JMAX-1)+U(I,JMAX-2)
          END IF
         ELSE
          DU=.5*(U(I,J+1)-U(I,J-1))
          DUU=U(I,J+1)-2.*U(I,J)+U(I,J-1)
         END IF
         IF (J.LE.2.OR.J.GE.JMAX-1) THEN
          IF (J.LE.2) THEN
          DUUU(J)=.5*(-3.*U(I,J+4)+14.*U(I,J+3)-24.*U(I,J+2)
     1                +18.*U(I,J+1)-5.*U(I,J))
          ELSE
           DUUU(J)=.5*(5.*U(I,J)-18.*U(I,J-1)+24.*U(I,J-2)
     1                -14.*U(I,J-3)+3.*U(I,J-4))
          ENDIF
        ELSE
       DUUU(J)=.5*(U(I,J+2)-2.*U(I,J+1)+2.*U(I,J-1)-U(I,J-2))
         ENDIF
         H3=H3+DUUU(J)**2
         H2= H2+DUU**2.
         HO= HO + DU**2.
         RMSDU= RMSDU + DU**2
         RMSDUU= RMSDUU + DUU**2.
         RMSDUUU=RMSDUUU+DUUU(J)**2
        IF(I.EQ.2.OR.I.EQ.IMAX)
     1    WRITE (6,200) J,DU,DUU,DUUU(J),DUUU(J)**2.
  200  FORMAT (1X,I5,4E13.5)
   20   CONTINUE
   10  CONTINUE
       H = WH3*H3 + WH2*H2 + WHO*HO
       RMSH=SQRT(H/(IMAX*JMAX))
       RMSDU=SQRT(RMSDU/(IMAX*JMAX))
       RMSDUU=SQRT(RMSDUU/(IMAX*JMAX))
       RMSDUUU=SQRT(RMSDUUU/(IMAX*JMAX))
       WRITE(6,300) H,RMSH,RMSDU,RMSDUU,RMSDUUU
  300  FORMAT(1X,'THE H=',E15.9,3X,'THE RMS OF H=',E13.5,/,
     1  1X,'THE RMS OF DU=',E13.5,3X,'THE RMS OF DUU=',E13.5,
     2  /,1X,'THE RMS OF DUUU=',E13.5,/)
       RETURN
       END
```

```
C   THIS SUBROUTINE CALCULATES THE POHLHAUSEN APPROXIMATE SOLUTION
C   FOR INCOMPRESSIBLE, FLAT PLATE.
C
      SUBROUTINE POHL(Y,DELTA,UE,IMAX,JMAX,UEXACT)
      DIMENSION Y(IMAX,JMAX),DELTA(IMAX),UEXACT(IMAX,0:JMAX+1)
      AMBDA=0.0
      A=0.0
      B=2.0+AMBDA/6.
      C=-AMBDA/2.
      D=-2.+AMBDA/2.
      E=1-AMBDA/6.
      DO 30 I=1,IMAX
       UEXACT(I,0)=0.0
 30   CONTINUE
      DO 10 I=2,IMAX
       DO 20 J=1,JMAX
        IF(Y(I,J).LE.DELTA(I))THEN
         UEXACT(I,J)=(A+B*(Y(I,J)/DELTA(I))+C*((Y(I,J)/DELTA(I))**2.)+
     1    D*((Y(I,J)/DELTA(I))**3.)+E*((Y(I,J)/DELTA(I))**4.))
        ELSE
         UEXACT(I,J)=UE/UE
        ENDIF
 20    CONTINUE
 10   CONTINUE
      DO 60 J=1,JMAX
       UEXACT(1,J)=UEXACT(2,J)
 60   CONTINUE
      RETURN
      END
C
C
C  THIS SUBROUTINE CALCULATES THE NEW GRID GENERATION CONTROL
C  FUNCTION P.  IT IS BASED ON A GLOBAL TRUNCATION ERROR
C  ANALYSIS FOR THE FLOW SOLUTION IN THE TRANSFORMED PLANE
C
      SUBROUTINE NEWP1 (U,P,UPLUSDC,UMINDC,DUUU,CPLUSDC,
     1            CMINDC,C,RMSDUUU,H,K,PERCENT,IMAX,JMAX,
     2      ABAR2,NMAX,X,Y,AAA,CCC,DDD,GGG,WWW,YMIN,YMAX,ICOL,
     3   ITERATE,UE,DELTA,P3,Y3,D3U,U3,A,ZI,IR,WHO,WH2,WH3)
      DIMENSION DUUU(JMAX),P(JMAX)
      DIMENSION UPLUSDC(IMAX,0:JMAX+1),UMINDC(IMAX,0:JMAX+1)
      DIMENSION U(IMAX,0:JMAX+1),U3(IMAX,0:JMAX+1)
      DIMENSION P3(JMAX),Y3(IMAX,JMAX),D3U(JMAX)
      DIMENSION X(IMAX),Y(IMAX,JMAX),DELTA(IMAX)
      DIMENSION ABAR2(NMAX),A(NMAX),ZI(NMAX,NMAX),WWW(JMAX)
      DIMENSION AAA(JMAX),CCC(JMAX),DDD(JMAX),GGG(JMAX)
      WRITE(6,100)
 100  FORMAT(/,15X,'******** NEW RLAMBA CALCULATION *********',//,
     1        3X,'ETA',5X,'DU',11X,'DUU',10X,
     2        'POLD',10X,'DUUU',7X,'DUUU**2')
        RMSDU=0.0
        RMSDUU=0.0
        RMSDUUU=0.0
        H2=0.0
        H3=0.0
        HP2=0.0
        HP3=0.0
        HM2=0.0
        HM3=0.0
        HP0=0.0
        HM0=0.0
        H0=0.0
             COLD=C
      DO 20 I=1,IMAX
C      I=ICOL
      IF(I.EQ.2.OR.I.EQ.IMAX)  WRITE(6,*)'I= ',I
       DO 10 J=1,JMAX
        IF (J.EQ.1.OR.J.EQ.JMAX) THEN
         IF (J.EQ.1) THEN
          DU=.5*(-3.*U(I,1)+4.*U(I,2)-U(I,3))
```

```
            DUP=.5*(-3.*UPLUSDC(I,1)+4.*UPLUSDC(I,2)-UPLUSDC(I,3))
            DUM=.5*(-3.*UMINDC(I,1)+4.*UMINDC(I,2)-UMINDC(I,3))
            DUU=U(I,1)-2.*U(I,2)+U(I,3)
            DUUP=UPLUSDC(I,1)-2.*UPLUSDC(I,2)+UPLUSDC(I,3)
            DUUM=UMINDC(I,1)-2.*UMINDC(I,2)+UMINDC(I,3)
           ELSE
            DU=.5*(3.*U(I,JMAX)-4.*U(I,JMAX-1)+U(I,JMAX-2))
            DUP=.5*(3.*UPLUSDC(I,JMAX)-4.*UPLUSDC(I,JMAX-1)+
     1           UPLUSDC(I,JMAX-2))
            DUM=.5*(3.*UMINDC(I,JMAX)-4.*UMINDC(I,JMAX-1)+
     1           UMINDC(I,JMAX-2))
            DUU=U(I,JMAX)-2.*U(I,JMAX-1)+U(I,JMAX-2)
            DUUP=UPLUSDC(I,JMAX)-2.*UPLUSDC(I,JMAX-1)+
     1           UPLUSDC(I,JMAX-2)
            DUUM=UMINDC(I,JMAX)-2.*UMINDC(I,JMAX-1)+
     1           UMINDC(I,JMAX-2)
           END IF
          ELSE
           DU=.5*(U(I,J+1)-U(I,J-1))
           DUP=.5*(UPLUSDC(I,J+1)-UPLUSDC(I,J-1))
           DUM=.5*(UMINDC(I,J+1)-UMINDC(I,J-1))
           DUU=U(I,J+1)-2.*U(I,J)+U(I,J-1)
           DUUP=UPLUSDC(I,J+1)-2.*UPLUSDC(I,J)+UPLUSDC(I,J-1)
           DUUM=UMINDC(I,J+1)-2.*UMINDC(I,J)+UMINDC(I,J-1)
          END IF
          IF (J.LE.2.OR.J.GE.JMAX-1) THEN
           IF (J.LE.2) THEN
            DUUU(J)=.5*(-3.*U(I,J+4)+14.*U(I,J+3)-24.*U(I,J+2)+
     1           18.*U(I,J+1)-5.*U(I,J))
            DUUUP=.5*(-3.*UPLUSDC(I,J+4)+14.*UPLUSDC(I,J+3)-
     1           24.*UPLUSDC(I,J+2)+18.*UPLUSDC(I,J+1)-5.*UPLUSDC(I,J))
            DUUUM=.5*(-3.*UMINDC(I,J+4)+14.*UMINDC(I,J+3)-
     1           24.*UMINDC(I,J+2)+18.*UMINDC(I,J+1)-5.*UMINDC(I,J))
           ELSE
            DUUU(J)=.5*(5.*U(I,J)-18.*U(I,J-1)+24.*U(I,J-2)-14.*U(I,J-3)+
     1           3.*U(I,J-4))
          DUUUP=.5*(5.*UPLUSDC(I,J)-18.*UPLUSDC(I,J-1)+24.*UPLUSDC(I,J-2)-
     1           14.*UPLUSDC(I,J-3)+3.*UPLUSDC(I,J-4))
            DUUUM=.5*(5.*UMINDC(I,J)-18.*UMINDC(I,J-1)+24.*UMINDC(I,J-2)-
     1           14.*UMINDC(I,J-3)+3.*UMINDC(I,J-4))
           END IF
          ELSE
           DUUU(J)=.5*(U(I,J+2)-2.*U(I,J+1)+2.*U(I,J-1)-U(I,J-2))
           DUUUP=.5*(UPLUSDC(I,J+2)-2.*UPLUSDC(I,J+1)+2.*UPLUSDC(I,J-1)-
     1           UPLUSDC(I,J-2))
           DUUUM=.5*(UMINDC(I,J+2)-2.*UMINDC(I,J+1)+2.*UMINDC(I,J-1)-
     1           UMINDC(I,J-2))
           END IF
         IF(I.EQ.2.OR.I.EQ.IMAX) THEN
          WRITE (6,200) J,DU,DUU,P(J),DUUU(J),DUUU(J)**2.
200      FORMAT (I5,5E13.5)
         ENDIF
         RMSDUUU=RMSDUUU+DUUU(J)**2
         RMSDU=RMSDU+DU**2
         RMSDUU=RMSDUU+DUU**2.
         HO=HO+DU**2.
         H2=H2+DUU**2.
         H3=H3+DUUU(J)**2
         HPO=HPO+DUP**2.
         HP2=HP2+DUUP**2.
         HP3=HP3+DUUUP**2.
         HMO=HMO+DUM**2.
         HM2=HM2+DUUM**2.
         HM3=HM3+DUUUM**2
10       CONTINUE
20       CONTINUE
         H=WH3*H3 + WH2*H2 +WHO*HO
         HPLUSDC=WH3*HP3 +WH2*HP2 +WHO*HPO
         HMINDC=WH3*HM3+WH2*HM2 + WHO*HMO
          DET=(CPLUSDC**2)*(C-CMINDC)-CPLUSDC*(C*C-CMINDC**2)+
```

```
     1        C*C*CMINDC-(CMINDC**2)*C
          RNUM=(CPLUSDC**2)*(H-HMINDC)-HPLUSDC*(C*C-CMINDC**2)+
     1         (C**2)*HMINDC-(CMINDC**2)*H
          DEN=HPLUSDC*(C-CMINDC)-CPLUSDC*(H-HMINDC)+H*CMINDC-
     1        HMINDC*C
          IF (DET.NE.0.0) THEN
           C=-.5*RNUM/DEN
           IF (DEN/DET.LT.0.0) THEN
            HMIN=MIN(HMINDC,H,HPLUSDC)
            IF (HMINDC.EQ.HMIN) C=CMINDC
            IF (HPLUSDC.EQ.HMIN)C=CPLUSDC
            IF(H.EQ.HMIN) C=COLD
           ELSE
            DO 40 JJ=1,NMAX
            ABAR2(JJ)=A(JJ)+ C * ZI(JJ,IR)
    40      CONTINUE
            CALL PCAL(ABAR2,NMAX,P3,JMAX)
            CALL EXPGRD(X,Y3,P3,AAA,CCC,DDD,GGG,WWW,YMIN,
     1       YMAX,IMAX,JMAX,ICOL)
            CALL BLIMP(X,Y3,IMAX,JMAX,U3,ITERATE,UE,DELTA,RE)
            CALL NWP(U3,D3U,H4,IMAX,JMAX,ICOL,WHO,WH2,WH3)
            HMIN=MIN(HMINDC,H,HPLUSDC,H4)
            IF(H4.EQ.HMIN.AND.H4.NE.H) GO TO 220
            IF (HMINDC.EQ.HMIN) C=CMINDC
            IF (HPLUSDC.EQ.HMIN)C=CPLUSDC
            IF(H.EQ.HMIN) C=COLD
    220    CONTINUE
           END IF
          ELSE
           HMIN=MIN(HMINDC,H,HPLUSDC)
           IF (HMINDC.EQ.HMIN) C=CMINDC
           IF (HPLUSDC.EQ.HMIN) C=CPLUSDC
           IF (H.EQ.HMIN)  C=COLD
          END IF
         EPSILON=.005
         IF (ABS(C).LT..01) EPSILON=.01
          DC=PERCENT*C+EPSILON
          CPLUSDC=C+DC
          CMINDC=C-DC
          WRITE (6,300) H,HMIN,HPLUSDC,HMINDC,C,K
    300   FORMAT (1X,'H=',E15.9,/,
     1          1X,'HMIN=',E15.9,/,
     2          1X,'HPL=',E15.9,/,
     3          1X,'HMN=',E15.9,/,
     4           1X,'NEW RLAMBA=',E15.9,/,
     5           1X,'ITERATION NUMBER, K=',I5,/)
          RMSDUUU=SQRT(RMSDUUU/(IMAX*JMAX))
          RMSDUU= SQRT(RMSDUU/(IMAX*JMAX))
          RMSDU= SQRT(RMSDU/(IMAX*JMAX))
          WRITE (6,400) RMSDU,RMSDUU,RMSDUUU,COLD
    400   FORMAT (1X,'RMSDU =',E13.5,3X,'RMSDUU =',E13.5,3X,'RMSDUUU='
     1          ,E13.5,/,1X,'RLAMBA OLD=',E15.9,/)
          RETURN
          END
```

Vita

Donald E. Coffey, Jr. was born on 21 July,1954 in Albuquerque, New Mexico. He graduated from Arlington Heights High School, Fort Worth, Texas in 1972. He attended the University of Texas one year before entering the US Air Force Academy in 1973. In 1977, he graduated from the US Air Force Academy with a Bachelor of Science degree in Aeronautical Engineering. He went through Undergraduate Pilot Training (UPT) at Laughlin AFB, Texas. From 1979 to 1984, he was assigned to the Ninth Bombardment Squadron at Carswell AFB, Texas, flying the Boeing B-52. In June 1984, he entered the School of Engineering at the Air Force Institute of Technology a Wright-Patterson AFB, Ohio.

Permanent Address: 14913 Murfin Rd.
Austin, Texas 78734

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A164038

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/GA/AA/85D-4	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: School of Engineering; 6b. OFFICE SYMBOL (If applicable): AFIT/ENY	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code): Air Force Institute of Technology, Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. — PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification): See Box 19	
12. PERSONAL AUTHOR(S): Donald E Coffey, Hr., B.S., Capt, USAF	
13a. TYPE OF REPORT: MS Thesis; 13b. TIME COVERED: FROM _____ TO _____; 14. DATE OF REPORT (Yr., Mo., Day): 1985 December; 15. PAGE COUNT: 150	
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
01	01	
20	04	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Adaptive Grids, Minimization Techniques, Boundary-Layer Codes, Truncation Error Mimimizing

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: ADAPTIVE-GRID OPTIMIZATION FOR MINIMIZING STEADY-STATE, TRUNCATION ERROR

Thesis Chairman: Sal A. Leone, Captain, USAF
Assistant Professor, Dept. of Aeronautics and Astronautics, Air Force Institute of Technology

Approved for public release: IAW AFR 190-1.
LYNN E. WOLAVER 16 JAN 86
Dean for Research and Professional Development
Air Force Institute of Technology (AFIT)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Dr. Sal A. Leone, Captain, USAF	22b. TELEPHONE NUMBER (Include Area Code): 513-255-6998; 22c. OFFICE SYMBOL: AFIT/ENY

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

This ~~study~~ thesis develops an adaptive-grid method which minimizes the truncation error in the finite-difference solution. The study solves compressible, steady-state, boundary-layer equations assuming perfect-gas flow over an isothermal wall. The Dorodnitsyn, compressibility transformation changes the boundary-layer equations, as expressed in two-dimensional, cartesian coordinates, into an incompressible form. The equations are then transformed into variables of a computational plane. Implicit Successive-Over-Relaxation (SOR) solves the finite-differenced, computational, boundary-layer equations. Comparison of the computed solution for incompressible flow over a flat plate to Blasius', exact solution shows the boundary-layer code is accurate.

The adaptive-grid method uses Powell's method to optimize the solution grid by minimizing the sum of the third derivative in the computational plane of the tangential velocity component. Powell's method finds the grid, control function, Q, in an elliptic, grid equation, $Y_{\eta\eta} + QY_{\eta} = 0$, which minimizes a specified function. The grid equation generates the grid spacing at the end of the plate. This spacing is then streamwise scaled across the remaining grid. Minimizing the sum of the square of the third derivative in the computational plane of the tangential velocity component, $U_{\eta\eta\eta}^2$, over the entire domain decreases the truncation error the best of the functions tested. This study tests the sum of the squares of the first, second, and third derivative of the tangential velocity as minimized functions. The accuracy of the optimized, adaptive-grid solution is greater than the original, fixed-grid solution. The study then applies the optimization to supersonic and hypersonic problems. The computed, adaptive-grid solutions show good correlation with theoretical models for supersonic and hypersonic flow developed by Van Driest and Eckert.